Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Aeronáuticos

Resolution of the Adjoint Navier-Stokes Equations using a Preconditioned Multigrid Method

Tesis Doctoral

Fernando Gisbert Cervera

Ingeniero Aeronáutico

Madrid, 2007

Departamento de Motopropulsión y Termofluidodinámica

Escuela Técnica Superior de Ingenieros Aeronáuticos

Resolution of the Adjoint Navier-Stokes Equations using a Preconditioned Multigrid Method

Autor

Fernando Gisbert Cervera

Ingeniero Aeronáutico

Director

Roque Corral García

Doctor Ingeniero Aeronáutico

Madrid, Enero 2007

Escuela Técnica Superior de Ingenieros Aeronáuticos

Universidad Politécnica de Madrid

Tribunal nombrado por el Magnífico y Excelentísimo Señor Rector de la Universidad Politécnica de Madrid. El día de de 2007.

Presidente:	D. Javier Jiménez Sendín
Vocal:	D. Jens Dominik Müller
Vocal:	D. Alfredo Pinelli
Vocal:	D. Carlos David Pérez Segarra
Secretario:	D. Juan Antonio Hernández Ramos
Suplente:	D. Michele Sergio Campobasso
Suplente:	D. Adel-Sayed Abbas Bayoumi

Realizado el acto de defensa y lectura de la tesis el día de de 2007 en la E.T.S.I. Aeronáuticos.

El Presidente Los Vocales

El Secretario

Aquesta tesi va dedicada a Mercedes i a Eliseo,

a Mercè i a Robert,

a Carlos i a Clara,

a Roc,

i a Ingrid.

Madrid, Març 2007

Agradecimientos

Esta tesis, que es producto de casi cinco años de trabajo, tiene contribuciones de mucha gente. Sin la ayuda de todos ellos, no me cabe la menor duda de que este trabajo no habría visto la luz.

En primer lugar, mi mayor gratitud es para Roque Corral, que me ha permitido desarrollar esta tesis con total libertad y confianza en mi trabajo y asistir a conferencias y cursos que han sido de gran valor para mí.

Quiero agradecer también a la gente del departamento de Tecnología y Métodos de ITP su ayuda. A Jaime Fernández-Castañeda, Jesús Contreras, y Carlos Vasco, por sus consejos y su paciencia mientras yo aprendía a manejarme y a comprender el Mu^2s^2T y el resto de herramientas del sistema de diseño. A Javier Crespo, porque sus trabajos en la paralelización del código eran requisito imprescindible para que los míos progresaran. Y en fin, a la gente del departamento, porque ellos han sido, son, y serán mis compañeros de trabajo, y porque todos, de una manera u otra, han hecho alguna contribución a este trabajo.

Agradezco también a la gente de Aerodinámica sus consejos, sobretodo en lo que al diseño de las paredes no axilsimétricas se refiere.

Por último, gracias a los miembros del tribunal por acceder a juzgar este trabajo.

Abstract

Resolution of the Adjoint Navier-Stokes equations using a Preconditioned Multigrid Method

Fernando Gisbert Cervera

Escuela Técnica Superior de Ingenieros Aeronáuticos. Madrid, 2007

The adjoint Navier-Stokes equations are solved by means of a preconditioned multigrid method. The use of this solver is motivated by the design improvements that may be obtained with a gradient-based optimisation method. The resolution of the adjoint Navier-Stokes equations yields significant computational savings, proportional to the number of optimisation variables when the gradient is evaluated. In the design of a turbine blade, where the definition of the geometry involves hundreds of geometric parameters, the savings are substantial.

The adjoint equations solver is derived from an existing parallel Reynolds-Averaged Navier-Stokes equations solver for unstructured grids, whose explicit integration scheme is enhanced with a preconditioned multigrid, providing rapid convergence for two and three-dimensional viscous cases. The coarse meshes of the multigrid method are obtained with an edge-based volume agglomeration approach. The preconditioned equations are analysed, with especial emphasis on the discretisation of the equations for stretched cells. It is demonstrated that the semi-coarsening of stretched cells yields a suitable agglomeration technique to improve the performance of the preconditioned multigrid method. The results of the linear analysis of the Navier-Stokes equations are valid for the adjoint provided both systems of equations have the same eigenvalues.

The adjoint solver is used in conjunction with a gradient-based optimisation to design the end walls of a low pressure turbine row using non-axisymmetric perturbations. The optimised solution yields subtantial reductions of the secondary kinetic energy with much less computational effort than the manual design.

Resumen

Resolución de las ecuaciones de Navier-Stokes adjuntas mediante un método multimalla precondicionado

Fernando Gisbert Cervera

Escuela Técnica Superior de Ingenieros Aeronáuticos. Madrid, 2007

Las ecuaciones de Navier-Stokes adjuntas se resuelven con un método multimalla precondicionado. La utilización de este resolvedor se fundamenta en las mejoras de diseño que se obtienen con un método de optimización basado en el gradiente. El ahorro de tiempo que se consigue cuando se utiliza la solución de las ecuaciones de Navier-Stokes adjuntas para calcular el gradiente es proporcional al número de grados de libertad del problema de optimización. La definición de la geometría de una fila de álabes de turbina contiene cientos de parámetros, por lo que el ahorro en tiempo de cálculo es sustancial.

El resolvedor de las ecuaciones de Navier-Stokes adjuntas se deriva de un resolvedor en paralelo de las ecuaciones promediadas de Navier-Stokes para mallas no estructuradas, cuyo esquema de integración explícito se ha mejorado con un método multimalla precondicionado. Con este método las ecuaciones de Navier-Stokes se convergen rápidamente al estado estacionario, tanto para casos bidimensionales como tridimensionales. Las mallas bastas del método multimalla se han generado mediante un método de aglomeración de volúmenes basado en las aristas de la malla. Se ha analizado la discretización resultante del método de precondicionado, con especial atención a la discretización de las ecuaciones en mallas con alargamiento. Se demuestra que la aglomeración direccional de las celdas alargadas, donde sólo aglomeramos en la dirección de menor alargamiento, es una técnica adecuada para mejorar las prestaciones del método multimalla precondicionado. Los resultados del análisis lineal de las ecuaciones de Navier-Stokes son también válidos para las ecuaciones adjuntas, ya que ambos problemas tienen los mismos autovalores.

El resolvedor de las ecuaciones adjuntas se ha utilizado conjuntamente con un método de optimización basado en el gradiente para diseñar las paredes de una fila de álabes de turbina de baja presión utilizando perturbaciones no axilsimétricas. El diseño obtenido reduce apreciablemente la energía cinética secundaria en un tiempo de cálculo menor que el que se emplea en la realización manual del diseño.

Contents

1. Introduction

The growth of the computational power during the last decades and its cost decrease have extended the use of computers to tackle complex engineering problems, such as the design of turbomachines. This task needs tools that generate the turbomachine geometry and grid it, other tools that simulate the physical behaviour of the generated geometry, and finally, others to post-process the results obtained from the simulations and to validate the proposed design according to multiple targets (efficiency, weight, noise, cost, etc.). Among all, the more time-consuming task is the fluid dynamics simulation, which involves the resolution of the Navier-Stokes equations.

Fluid Mechanics is a multiscale problem, ranging from the airfoil to the Kolmogorov scale, where the dissipation of the kinetic energy occurs. The accurate solution of all the scales requires a huge amount of computational power, far beyond the possibilities of nowadays supercomputers. Thus, the Navier-Stokes equations are simplified to obtain a problem that can be simulated within a reasonably amount of time. Today, the most common simplification consists in the modellisation of the smallest scales, drastically reducing the number of necessary points to discretise the simulated domain. The Euler and the Reynolds-Averaged Navier-Stokes (RANS) equations, or the LES models, are approximations to the Navier-Stokes equations intended to achieve relatively accurate results at affordable computation costs. The RANS equations together with a turbulence model [7, 86] provide the best balance between computation cost and solution accuracy with the current CPU resources. Even within this approach, the spatial resolution is quite often marginal and the use of finer grids leads to an improvement of the solution quality. This situation explains why the methods to solve the Navier-Stokes equations have been continuously improved in parallel with the computer performances. Implicit solvers, Runge-Kutta schemes, residual smoothing, preconditioning techniques, multigrid methods, etc., are algorithm developments that either alone or in conjunction with parallel computing techniques, accelerate the resolution of the flow governing equations [41, 46, 55, 61, 18, 37, 19]. With their use, the CPU cost of solving a given problem scales linearly with the number of grid points used to discretise

Year	Machine	Time/iteration (s)
2001	Pentium III @ 500 MHz	20.27
2006	Pentium IV @ 3 GHz	3.2

Table 1.1: Time per iteration with the baseline 3D unstructured solver for a 191000 points mesh in years 2001 and 2006.

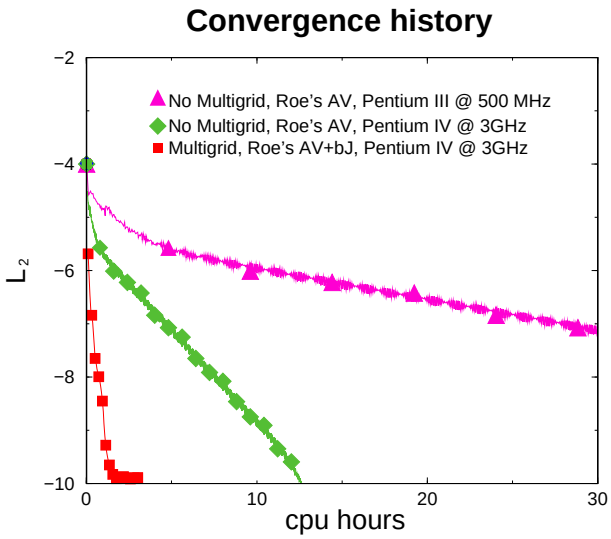


Figure 1.1: Comparison of times needed to reach a steady solution, using different computers and different resolution algorithms.

the domain.

Table 1.1 shows the reduction in computational time due to the hardware evolution. It is observed that the time employed to perform a single iteration in an unstructured RANS solver [37, 19] has been divided by about six in five years due to the use of newer machines. The speed-up in this case is directly associated to the increase in clock speed. However, the number of iterations required to reach the steady state convergence is the same in both simulations, since the resolution algorithm has not been changed. The effect of introducing additional algorithm improvements is shown in figure 1.1. The difference in computational time between a simulation carried in a Pentium III @ 500 MHz and a Pentium IV @ 3 GHz for the same case presented in table 1.1 is shown. If we add efficient convergence acceleration techniques to the baseline solver, such as a preconditioned multigrid method, the steady state is reached about ten times faster. Hence, improvements on the resolution algorithms may have an impact in the reduction of the CPU time as large as the use of more powerful machines. The combination of both effects produces a convergence acceleration factor of nearly sixty with respect to the same case simulated in 2001, which supposes a great advance in the simulation capabilities.

Another approach to reduce the design time consists in the automatic optimisation of the design parameters. The optimisation method links the geometry generation, the grid generation,

the resolution of the Navier-Stokes equations and the post-processing of the solution to obtain a design target value. That value is the input for the optimiser, that analyses it and produces a new set of design parameters as an output, starting the whole design cycle again. The optimisation could be performed with an evolutionary or genetic algorithm or with a gradient-based optimiser. The former is powerful when finding a global extremum, and works well when dealing with noisy target functions, but requires a large amount of CPU. With the latter, the obtention of the global extremum is not ensured, but consumes far less CPU time, especially if an efficient method to compute the gradient exists. Adjoint methods fulfil this demand. They have been used for the aerodynamic optimisation of airfoils, wings, and even complete aircrafts in stationary conditions [47, 44, 49], and also for the minimisation of the high cycle fatigue effects in rotor blades under periodic aerodynamic excitations [27], with satisfactory results in both cases.

The use of a gradient based optimiser together with an adjoint Navier-Stokes solver in the turbomachinery design could shorten the design cycle time. Moreover, combined with the convergence acceleration techniques mentioned above, it turns out to be a tool that could help to improve turbine designs at a very reasonable cost. We will show the advantages of using such a technique in the design of the end walls of a turbine row.

1.1. Multigrid

The size of the system of equations resulting from the discretisation of the Navier-Stokes equations often restricts the use of implicit solvers due to the large memory requirements. In that sense, explicit solvers, such as Runge-Kutta methods, are less restrictive and also easier to parallelise. Their main drawback is the time step limitation inherent to the resolution method, that slows the convergence to the steady state, hence becoming unfeasible in a design environment. Convergence acceleration algorithms try to minimise the CPU impact of choosing an explicit solver without assuming the memory limitations of an implicit scheme. Among them, the multigrid method is the most powerful one. With the use of a multigrid algorithm, convergence rate becomes mesh-independent, i.e.: the steady state can be reached with a number of operations proportional to the number of unknowns despite the grid size.

Multigrid methods first appeared in the 1960s, when Fedorenko [29] and Bakhvalov [6] proved the convergence properties of the method for elliptic operators. They proved that such operators could be converged with a number of operations $\mathcal{O}(N)$, being N the number of grid points, but the constant which affected N was so large that they thought that other convergence methods

were more efficient. Later in the late 1970s, Brandt [10] and others [70, 38] demonstrated the convergence properties of the multigrid method for solving certain elliptic PDEs. Even though there is little theory for the multigrid methods applied to the resolution of hyperbolic equations and their efficiency is much lower than for elliptic equations [64], their use has grown very much since their first use in the CFD field [69, 41, 46, 56].

For elliptic equations, the multigrid convergence properties rely on the ability of the iterative scheme to damp high frequency errors. Thus, the multigrid algorithm tries to convert the errors that are smooth on a fine grid into oscillatory errors in a coarser one, hence taking advantage of the good damping properties for oscillatory errors in every grid level. Textbook multigrid efficiency, i.e., the convergence of the problem in less than ten work units, where a work unit is defined as the cost of performing a relaxation step in the fine grid, has been attained for elliptic equations with appropriate smoothing algorithms.

In the context of CFD, multi-stage Runge-Kutta schemes are standard relaxation methods for explicit solvers [41, 55, 37]. For appropriate values of the time step and the numerical dissipation, the iterative scheme fulfils the multigrid requirement that high frequency errors are effectively damped. That is the reason why the multigrid technique is also useful for hyperbolic equations. Due to the spatial discretisation, there are error modes that are not convected, but only damped. These errors are treated as they would in the resolution of elliptic PDE, and therefore multigrid is very effective in damping them. For the rest of errors that are both damped and propagated, the multigrid accelerates their propagation in the coarser grids, therefore they are expelled out of the domain faster.

Obtaining multiple grid levels for the multigrid method is straightforward in structured meshes, eliminating one of every two nodes in each direction. However, the coarsening is a key issue in unstructured grids. There are two common techniques to coarsen an initial fine grid, namely edge collapsing [66, 65, 85] and volume agglomeration [62, 59]. In the former, high quality coarse meshes are obtained imposing geometric restrictions in the generated elements. The drawback of the method is that the restrictions often prevent new elements from being created and hence the coarsening ratios decrease when coarser levels are constructed. The volume agglomeration approach is simpler, and consists in fusing the neighbouring volumes of a seed volume to form an agglomerated volume. However, the control of the geometry in the coarse grids is poor and may spoil the discretisation of viscous terms. Perhaps the main advantage of this method is that it can be entirely edge-based, thus the coarsening algorithm is transparent

to the type of elements used in the initial mesh. Additional geometry control is often included to preserve the quality of the mesh in zones where the pure application of the basic agglomeration algorithm generates excessively degraded elements, such in stretched boundary layer cells [60], in semi-unstructured meshes or in grids made up of quads.

The multigrid method remains efficient as long as the Runge-Kutta succeeds in damping high frequency errors. However, this situation is not guaranteed when solving the RANS equations. The problem is presented when discretising the boundary layer region, by placing much more points in the normal than in the streamwise direction, which results in highly stretched cells that increase the stiffness of the discrete system of equations. These cases prevent the Runge-Kutta scheme to damp, and even to propagate, most of the errors, drastically degrading the multigrid convergence rate. Most of these problems may be fixed using local preconditioning.

1.2. Local preconditioning

Preconditioning techniques modify some of the spatial eigenvalues of the problem to place the high frequency error modes in regions of the Runge-Kutta stability map where they can be appropriately damped. Its use was pioneered by Chorin [15], who used a pseudo-compressibility to solve incompressible flows. Later, these methods were generalised for the resolution of low-Mach number problems [82, 51, 83]. Regions of low speed flow are encountered in several gas turbine applications: pressure side bubbles, disk cavities, massively separated regions, etc. In these regimes, the acoustic waves propagate essentially at the speed of sound c, while the entropy and vorticity waves are convected at the flow speed, which is much smaller. The large disparity in the eigenvalues causes the problem to be stiff. The low Mach number preconditioning method consists in premultiplying the time derivatives by a matrix that equals the propagation speed of all waves. Besides, the numerical dissipation must be modified to correct the asymptotic system behaviour when the Mach number tends to zero [84].

The block-Jacobi preconditioning method corrects the lack of damping for the high frequency modes in the highly stretched grids resulting from the discretisation of boundary layers at high Reynolds numbers [72]. Its use, either alone or in conjunction with the low Mach number preconditioning, produces an appropriate damping of almost all high frequency waves, for all the flow regimes and flow discretisation types, thus enhancing the multigrid smoother. If not properly damped, high frequency errors turn smooth in the coarser grid levels, a phenomenon known as aliasing, the efficiency of the relaxation scheme to damp them is very low and the

multigrid loses all its effectiveness. The analysis of the preconditioned spatial discretisation is also necessary to establish an appropriate multigrid strategy. If the preconditioned smoother is not able to damp certain high frequency eigenmodes, it is mandatory not to convert them into smooth modes when transferring the error to the coarse grid. This is the reason why semi-coarsening is used. It consists in coarsening the original mesh just in the direction of low stretching, hence avoiding the aliasing of high frequency errors in the direction of high stretching. The semi-coarsening technique enhances the convergence rate of the multigrid when combined with the block-Jacobi preconditioning. Analogously, the full coarsening does not produce satisfactory results [74].

1.3. Aerodynamic Optimisation using Adjoint Navier-Stokes Solvers

The algorithms outlined in the previous sections allow rapid evaluations of steady state flows around complex engineering configurations, hence reducing the design cycle time and allowing greater design refinement. However, obtaining the best possible design would require the use of an optimisation technique.

Manual optimisations, although possible, are not desirable, especially for the designer, since it is a tedious task to manually generate the geometries, solve the RANS equations, post-process the solution to obtain the best-possible design, and start the process again until convergence is reached. The advantage of this approach is that the designer can manage a great amount of information to decide whether a design is acceptable or not. The final solution contains all the designer's specifications, even qualitative details of the flow which are difficult to formulate explicitly, and hence they cannot be introduced in an automatic optimisation. Besides, the designer acquires some degree of knowledge about the problem which is solving while performing the manual optimisation, and this can be used to tackle future designs with an a priori know-how that is often hard to transmit to an algorithm. The use of an optimiser as a black box, without interaction with the designer, must be dismissed and, if used, must only serve to prospect innovative designs and to gain additional knowledge of the governing physics of the problem. Therefore, prior to the use of the optimiser as a valid design tool, the previously acquired knowledge must be systematised and incorporated to the optimisation algorithm. However, once these drawbacks have been overcome, the use of an optimiser is preferred, since the design cycle is considerably shortened and hence the designer can explore a greater number of solutions. The next step is to choose an optimisation algorithm.

As mentioned above, evolutionary algorithms lead to global optimums and are the preferred approach to prospect the whole design space, but the cost to perform such prospection becomes prohibitive as the number of parameters grows. Even though there are methods to guess how the design space looks like, e.g., neural networks with a large database of previously generated designs [9, 26, 31, 30], these methods often need a mature solver that does not modify the steady solution very much. Every modification in the accuracy of the RANS solver would suppose rerunning the whole design database, which seems impractical with the available computing resources.

On the other side, gradient-based algorithms provide only local information, thus the design space is not completely mapped and the global optimum is not ensured. However, these methods contain more intelligence, i.e. the use of the gradient information for each design point, and therefore the number of iterations to reach the same optimum is considerably smaller. Another tool makes the use of gradient-based algorithms more attractive than evolutionary algorithms: the adjoint solver. The use of aerodynamic adjoint solvers was pioneered by Pironneau [76] and later introduced by Jameson [42, 45, 44], who used it first for two-dimensional airfoil optimisations, then for wings and finally for complete aircraft configurations. It has also been used in turbomachinery to drive noise and flutter response optimisations [13, 27]. The adjoint formulation of the cost function gradient shows a very smart way of evaluating it without having to compute as many solutions as design parameters. By solving the adjoint equations of the governing physics of the optimisation problem the cost of evaluating the gradient is roughly equivalent to one cost function evaluation, therefore the speed-up of the gradient-based algorithm when using an adjoint solver is proportional to the number of design parameters.

Constructing the adjoint equations requires the linearisation of the Navier-Stokes equations and the formulation of their adjoint counterpart. The final result is a set of discrete equations, but there are basically two paths to reach that objective. One can discretise the Navier-Stokes equations, linearise them and then construct the discrete adjoint equations, or alternatively, one can linearise the Navier-Stokes PDEs, construct the adjoint PDEs and then discretise them. The former is the discrete approach to obtain the Navier-Stokes [34, 67, 71], and the latter is known as the continuous approach [45]. Each one has its pros and cons but the results obtained from both do not differ substantially [68]. However, before the development of this work, a code to solve the linearised discrete Navier-Stokes equations already existed. Constructing the discrete adjoint code from it makes the debugging of the adjoint code easier, since the gradient calculated

with the linear and adjoint codes must have exactly the same value after performing the same number of iterations. Achieving this goal requires the construction of the adjoint counterparts of all the linear operators used when solving the discrete linear Navier-Stokes equations without incurring in excessive memory or CPU time penalties.

The optimiser, together with the adjoint Navier-Stokes equations solver, must be inserted in an optimisation algorithm that includes the geometry and mesh generation, the simulation of the resulting domain with the RANS and adjoint codes and the post-processing to obtain the cost function and the gradient values. The resulting system manages the input and output data of all the programs involved in the design process. We have used it to design the end-walls of turbine rows [22, 23].

1.4. Summary

This thesis is divided into two parts. The former expounds the convergence acceleration algorithms implemented in the baseline solver: a multigrid method and a block-Jacobi preconditioning. The second part explains the steps followed to implement an adjoint Navier-Stokes solver, and its application to the optimisation of profiled end walls.

The second chapter presents the baseline solver, known as Mu^2s^2T [19], where all the modifications explained in the previous sections are implemented, paying attention to its peculiarities. It is a three-dimensional unstructured solver that uses an edge-based data structure to store the necessary data to address the resolution of the Reynolds-Averaged Navier-Stokes equations using a finite-volume discretisation.

The third chapter describes the fundamentals of the multigrid method for both elliptic and hyperbolic problems. Then its application to the resolution of the Navier-Stokes equations using unstructured meshes is discussed. The method to construct the coarser meshes, consisting in a fully edge-based agglomeration algorithm, is explained in detail in the fourth chapter. Special agglomeration features, such as the stretched cells treatment, the quadrilateral elements agglomeration and the three-dimensional agglomeration of semi-unstructured meshes, are also presented.

The fifth chapter is dedicated to analyse the effects of the numerical stiffness in the convergence rate. The discrete stiffness associated to the high stretching of the mesh cells is addressed analysing the spatial eigenvalues of the linearised Navier-Stokes equations. It will be seen that the stretching pushes the eigenvalues into zones where the Runge-Kutta scheme is not able to

damp the errors. It will be also demonstrated the convenience of using the matricial form of the numerical diffusion to avoid the excess of dissipation in stretched cells. However, using the matricial dissipation worsens the damping properties of high frequency errors, and the use of a preconditioning method becomes mandatory. The use of the block-Jacobi preconditioning solves the problems related with the lack of damping for almost every high frequency error. The eigenvalues that are not appropriately treated are excluded from the multigrid strategy, meaning that they are always solved as high frequency modes, even in the coarser grid levels. By applying the results of the analysis to the implementation of the preconditioned multigrid, the convergence rate of the discrete RANS equations is greatly improved for high values of the cell stretching, and the multigrid convergence properties become (almost) mesh-independent again. The effect of combining the block-Jacobi and the low Mach number preconditioning methods is also addressed, and it is shown that the use of such combination solves the problems associated with the physical stiffness derived from low Mach number conditions.

The sixth chapter describes the Dual Time Step method to solve unsteady cases with arbitrarily large time steps and the modifications that this method implies in both the computation of the preconditioning matrices and the eigenvalues of the resulting system of equations.

The convergence rate acceleration derived from the use of the preconditioned multigrid method is demonstrated in the seventh chapter for the Euler and the RANS equations in two-dimensional and three-dimensional simulations. The independence of the convergence rate from the grid spacing is also demonstrated. We also show results for unsteady cases where the multigrid has been combined with the dual time step method [43, 5].

The second part of the thesis is devoted to the derivation of the discrete adjoint Navier-Stokes equations, its implementation and finally its validation. Taking advantage of this tool, a gradient-based optimisation algorithm is used to minimise the secondary losses of turbine rows. The discrete adjoint Navier-Stokes equations are derived from the resolution of a Lagrange multipliers problem. Based on that derivation, the discrete adjoint operators for the viscous, inviscid, and artificial dissipation fluxes are obtained. To ensure the exact matching of the cost function derivatives computed with the adjoint solver with those obtained with the linearised Navier-Stokes equations, the adjoint iterative procedures are also developed, even though this step is not strictly necessary in terms of convergence properties of the adjoint equations. Thus, the adjoint counterparts of the multi-stage Runge-Kutta scheme, residual smoothing, block-Jacobi preconditioning and multigrid operators are also obtained. The validation of the adjoint

Navier-Stokes solver consists in checking that the gradient value obtained with the adjoint solver and by finite difference of two RANS solutions coincide. Once validated, the adjoint solver is inserted into an optimisation algorithm and used to design the end walls of turbine rows. The design is intended to minimise the secondary kinetic energy downstream the row. Different solutions will be presented and compared with manual designs. The limitations that arise when performing the automatic optimisation will also be shown.

Finally, three appendices are included. One with the formulation of the matrices involved in the block-Jacobi preconditioning method, either alone or combined with the low Mach number preconditioning method. Another with the obtention of the discrete adjoint Navier-Stokes equations, and the latter with the derivation of the target function to obtain its gradient.

2. 3D Unstructured Base solver

The aim of this chapter is to provide a brief introduction to the main features of an existing Reynolds-Averaged Navier-Stokes equations solver, which is the result of several works, mainly due to Dr. Gómez and also to the personnel of the Technology and Methods Department at ITP [18, 37, 16, 21, 17]. The developments presented in this thesis, especially those related with preconditioning and multigrid, are based on this previously existing solver.

The equations are solved in arbitrary domains which are meshed using unstructured grids. The data structure used to store the grid and the solution strongly influences the development of the algorithms that are presented along this work. In the next sections, the discretisation of the Navier-Stokes PDEs in unstructured meshes and the edge-based data structure used to store the necessary data to solve them will be outlined. Then the explicit time marching algorithm and the residual smoothing technique are presented. Finally, the turbulence models used in the solver will be introduced.

2.1. Reynolds-Averaged Navier-Stokes equations

The differential form of the Navier-Stokes equations is:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{f}}{\partial x} + \frac{\partial \mathbf{g}}{\partial y} + \frac{\partial \mathbf{h}}{\partial z} = \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} + \frac{\partial \mathbf{H}}{\partial z} \tag{2.1}$$

being $\mathbf{U} = [\ \rho \quad \rho u \quad \rho v \quad \rho w \quad \rho E\ ]^T$ the conservative variables,

$$[\ \mathbf{f} \quad \mathbf{g} \quad \mathbf{h}\ ] = \begin{bmatrix} \rho u & \rho v & \rho w \\ \rho u^2 + p & \rho uv & \rho uw \\ \rho uv & \rho v^2 + p & \rho vw \\ \rho uw & \rho vw & \rho w^2 + p \\ u\,(\rho E + p) & v\,(\rho E + p) & w\,(\rho E + p) \end{bmatrix} \tag{2.2}$$

the inviscid fluxes and

$$[\ \mathbf{F}\quad \mathbf{G}\quad \mathbf{H}\ ] = \begin{bmatrix} 0 & 0 & 0 \\ \tau_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{xy} & \tau_{yy} & \tau_{yz} \\ \tau_{xz} & \tau_{yz} & \tau_{zz} \\ u\tau_{xx}+v\tau_{xy}+w\tau_{xz}-q_x & u\tau_{xy}+v\tau_{yy}+w\tau_{yz}-q_y & u\tau_{xz}+v\tau_{yz}+w\tau_{zz}-q_z \end{bmatrix}$$

the viscous terms. As we use the perfect gas assumption, the gas state equation $p = \rho R_g T$ holds. Besides,

$$E = C_v T + \frac{1}{2}q^2,$$

where $q^2 = u^2 + v^2 + w^2$, and C_v is the specific heat at constant volume. The pressure is expressed as a function of the conservative variables:

$$p = \rho\left(\gamma - 1\right)\left(E - \frac{1}{2}q^2\right),$$

where $\gamma = C_p/C_v$ is the relation between the specific heats, at constant pressure and constant volume. The viscous stress tensor expression is

$$\tau_{ij} = \mu\left(\partial_i v_j + \partial_j v_i\right) - \frac{2}{3}\mu\left(\nabla\cdot\mathbf{v}\right)\delta_{ij} \tag{2.3}$$

and the heat flux vector is expressed using the Fourier's law:

$$\mathbf{q} = -k\nabla T \tag{2.4}$$

In the preceeding equations (2.3) and (2.4) the constants μ and k are the viscosity and the conductivity, respectively:

$$\begin{aligned} \mu &= \mu_l + \mu_t \\ k &= \left(\frac{\mu_l}{Pr_l} + \frac{\mu_t}{Pr_t}\right)\cdot C_p \end{aligned}$$

where the subindexes l and t stand for laminar and turbulent. The expression of the laminar viscosity is given by the Sutherland law

$$\mu_l = \frac{1.458 \cdot 10^{-6} T^{3/2}}{T + 110.4}$$

and the values of the laminar and turbulent Prandtl numbers for air are 0.72 and 0.9 respectively.

The equation (2.1) may be written in compact form

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}_c = \nabla \cdot \mathbf{F}_v$$

where $\mathbf{F}_c = \mathbf{f}\,\vec{\imath} + \mathbf{g}\,\vec{\jmath} + \mathbf{h}\,\vec{k}$ and $\mathbf{F}_v = \mathbf{F}\,\vec{\imath} + \mathbf{G}\,\vec{\jmath} + \mathbf{H}\,\vec{k}$. By integrating in a control volume ϑ with an associated boundary surface Σ, we obtain, applying the Gauss divergence theorem

$$\frac{d}{dt}\int_{\vartheta} \mathbf{U} d\vartheta + \int_{\Sigma} (\mathbf{F}_c - \mathbf{F}_v) \cdot \mathbf{n} d\sigma = 0 \tag{2.5}$$

where the products $\mathbf{F}_c \cdot \mathbf{n}$ and $\mathbf{F}_v \cdot \mathbf{n}$ are expressed as

$$\mathbf{F}_c \cdot \mathbf{n} = \begin{bmatrix} \rho v_n \\ \rho u v_n + p \cdot n_x \\ \rho v v_n + p \cdot n_y \\ \rho w v_n + p \cdot n_z \\ (\rho E + p)\, v_n \end{bmatrix}, \qquad \mathbf{F}_v \cdot \mathbf{n} = \begin{bmatrix} 0 \\ \tau_{xn} \\ \tau_{yn} \\ \tau_{zn} \\ \mathbf{v}^T \cdot \boldsymbol{\tau} \cdot \mathbf{n} - \mathbf{q} \cdot \mathbf{n} \end{bmatrix} \tag{2.6}$$

In these expressions, $v_n = \mathbf{v} \cdot \mathbf{n}$, and $\tau_{kn} = \tau_{kx} \cdot n_x + \tau_{ky} \cdot n_y + \tau_{kz} \cdot n_z$, where $k = x, y, z$.

2.2. Spatial discretisation

The solver uses hybrid unstructured grids to discretise the entire spatial domain with a single homogeneous data structure, in contrast with block-structured methods. This homogeneity eases the parallelisation, even though the indirect addressing of the data resulting from the use unstructured grids must be carefully treated. The construction of coarser meshes from an unstructured initial mesh becomes difficult, too.

The geometries of the axial gas turbine blades are essentially obtained by extruding and deforming a two-dimensional section of the blade. This property is usually inherited by the grids used to discretise the blade row, and therefore it is usual to mesh a two-dimensional

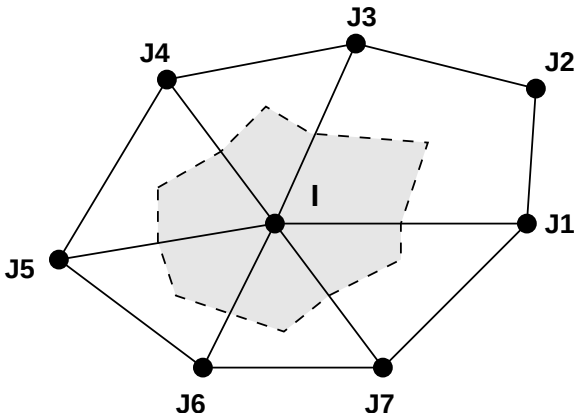


Figure 2.1: Hybrid cell grid and associated dual mesh.

section of the domain and then it is extruded and adapted to the three-dimensional variations of the blade geometry. Two-dimensional grids are generated creating highly stretched quads in the blade boundary layer by means of an advancing front algorithm, while the core flow region is meshed with triangles using a Delaunay approach [21]. The corresponding three-dimensional grid is generated by extruding the two-dimensional mesh along the three-dimensional boundaries of the domain that is simulated [17]. The resulting mesh contains hexahedra and prisms.

The discrete equations are obtained from Eq. (2.5) using a finite volume formulation, where the control volumes are constructed making use of the dual mesh (see figure 2.1), that is, the mesh that results from connecting the centroids of the cells that surround a node [37]. Thus, the solution is stored in the cell vertex, and the fluxes over the control volume faces are obtained computing the semi-sum of the fluxes in each node, which yields a second-order cell-centered scheme in the dual mesh. Hence the semi-discrete equivalent of Eq. (2.5) for each node i of an unstructured mesh is:

$$\frac{d\left(\mathbf{U}_i \cdot \vartheta_i\right)}{dt} + \sum_{j=1}^{\#ed_i} \frac{1}{2}\left[\left(\mathbf{F}_c - \mathbf{F}_v\right)\big|_i + \left(\mathbf{F}_c - \mathbf{F}_v\right)\big|_j\right] \cdot \mathbf{n}_{ij}\sigma_{ij} + \sum_{j=1}^{\#Bed_i} \left(\mathbf{F}_c - \mathbf{F}_v\right)\big|_j \cdot \mathbf{n}_{ij}\sigma_{ij} = 0 \quad (2.7)$$

where ϑ_i is the control volume of the dual cell i, σ_{ij} represents the area associated to the edge ij that joins the node i with its neighbours, and $\#ed_i$ and $\#Bed_i$ are the number of edges and domain boundary edges that reach that node. Since $\mathbf{n}_{ij} = -\mathbf{n}_{ji}$, the contribution to the fluxes of node j associated to the edge ij is opposite to that of node i. This result makes possible to evaluate the fluxes for all grid nodes by running a single loop over the edges of the mesh.

2.2.1. Evaluation of the inviscid and artificial viscosity terms

High frequency errors are neither damped nor propagated in a central difference scheme like that of Eq. (2.7)., and some amount of numerical diffusion is required to stabilise the method. The expression of the inviscid and artificial viscosity fluxes is, for an edge ij:

$$\frac{1}{2}\left(\mathbf{F}_c|_i + \mathbf{F}_c|_j\right)\cdot\mathbf{n}_{ij}\sigma_{ij} + \frac{1}{2}\left|A_{ij}\right|\Delta\mathbf{U} \tag{2.8}$$

where the matrix A_{ij} is the jacobian matrix of the inviscid fluxes:

$$A_{ij} = \frac{\partial\left[\left(\mathbf{F}_c|_i + \mathbf{F}_c|_j\right)\right]}{\partial\mathbf{U}}\cdot\mathbf{n}_{ij}\sigma_{ij}$$

If we diagonalise this matrix we can decompose it into a product

$$A_{ij} = T_{ij}\Lambda_{ij}T_{ij}^{-1}$$

where Λ_{ij} is a diagonal matrix whose elements are the eigenvalues of the jacobian matrix:

$$\Lambda_{ij} = \begin{bmatrix} \mathbf{v}_{ij}\cdot\mathbf{n}_{ij} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{v}_{ij}\cdot\mathbf{n}_{ij} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{v}_{ij}\cdot\mathbf{n}_{ij} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{v}_{ij}\cdot\mathbf{n}_{ij} + c_{ij} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{v}_{ij}\cdot\mathbf{n}_{ij} - c_{ij} \end{bmatrix}$$

being

$$\mathbf{v}_{ij} = \frac{1}{2}\left(\mathbf{v}_i + \mathbf{v}_j\right)$$

The matrices T_{ij} and T_{ij}^{-1} are written in appendix A. We define the matrix $|\Lambda_{ij}|$ as a diagonal matrix whose elements are the absolute values of Λ_{ij}, and therefore the matrix $|A_{ij}|$ is

$$\left|A_{ij}\right| = T_{ij}\left|\Lambda_{ij}\right|T_{ij}^{-1} \tag{2.9}$$

Depending on the choice of the matrix $|A_{ij}|$ and of the term $\Delta\mathbf{U}$ we may obtain different formulations of the artificial viscosity.

With regard to the choice of $|A_{ij}|$, the simplest formulation consists in substituting the absolute

values of $|\Lambda_{ij}|$ by the largest of them, hence simplifying the Eq. (2.9):

$$|A_{ij}| = (|\mathbf{v}_{ij} \cdot \mathbf{n}_{ij}| + c_{ij})\, I$$

being I the 5×5 identity matrix. This approach is known as scalar artificial dissipation, in contrast with the matricial artificial dissipation where the original Eq. (2.9) is maintained, and damps all error modes with the same damping coefficient $|\mathbf{v}_{ij} \cdot \mathbf{n}_{ij}| + c_{ij}$. Its implications on the convergence rate of the system of equations and on the quality of the solution will be discussed later in chapter 5.

2.2.1.1. Artificial viscosity construction using the pseudo-Laplacian

The value $\Delta\mathbf{U}$ can be a blend of second and fourth differences [48]:

$$\Delta\mathbf{U} = S_{ij}\,(\mathbf{U}_i - \mathbf{U}_j) - \frac{1}{2}\,(1-\kappa)\,(1-S_{ij})\,[L_i\,(\mathbf{U}) - L_j\,(\mathbf{U})] \tag{2.10}$$

The expression $L_i\,(\mathbf{U})$ stands for the pseudo-Laplacian operator:

$$L_i\,(\mathbf{U}) = \frac{1}{\#ed_i} \sum_{j=1}^{\#ed_i} (\mathbf{U}_j - \mathbf{U}_i) \tag{2.11}$$

and the value S_{ij} is a switch between the contribution of the pseudo-Laplacian terms and the sole difference of the conservative variables. This switch is used to avoid spurious oscillations of the solution in the vicinity of shock waves, where there are abrupt changes in the solutions. In such regions, the fourth differences are disabled and only the second ones are used, hence switching the discretisation to first order. This switch is based on the relation between the values of the pseudo-Laplacian of the pressure and the pressure itself,

$$S_{ij} = min\left[1, \varepsilon \cdot \left(\frac{|L_i\,(p)|}{p_i} + \frac{|L_j\,(p)|}{p_j}\right)\right] \tag{2.12}$$

where ε is a constant which is set to 8. Finally, κ is a constant to scale the fourth differences, which is set to $\kappa = 2/3$ throughout this work, yielding appropriate values for the damping of high frequency error modes.

2.2.1.2. MUSCL approach

Otherwise, the inviscid and the artificial viscosity fluxes can be evaluated using an upwind-biased interpolation of the solution [53]. Eq. (2.8) is evaluated with the variables $\mathbf{U}_i^+$ and $\mathbf{U}_j^-$, yielding

$$\frac{1}{2}\left(\mathbf{F}_c\left(\mathbf{U}_i^+\right)+\mathbf{F}_c\left(\mathbf{U}_j^-\right)\right)+\frac{1}{2}\left|A_{ij}\left(\mathbf{U}_i^+,\mathbf{U}_j^-\right)\right|\left(\mathbf{U}_i^+-\mathbf{U}_j^-\right)$$

where the variables in the edge nodes are defined by:

$$\begin{aligned}\mathbf{U}_i^+ &= \mathbf{U}_i+\frac{S_i}{4}\left[\left(1-\kappa_2 S_i\right)\boldsymbol{\Delta}_i^-+\left(1+\kappa_2 S_i\right)\left(\mathbf{U}_j-\mathbf{U}_i\right)\right]\\ \mathbf{U}_j^- &= \mathbf{U}_j-\frac{S_j}{4}\left[\left(1-\kappa_2 S_j\right)\boldsymbol{\Delta}_j^++\left(1+\kappa_2 S_j\right)\left(\mathbf{U}_j-\mathbf{U}_i\right)\right]\end{aligned}$$

being $\kappa_2 = 1/3$. The operators $\boldsymbol{\Delta}_i^-$ and $\boldsymbol{\Delta}_j^+$ are:

$$\begin{aligned}\boldsymbol{\Delta}_i^- &= 2\nabla\mathbf{U}_i\cdot\left(\mathbf{x}_j-\mathbf{x}_i\right)-\left(\mathbf{U}_j-\mathbf{U}_i\right)\\ \boldsymbol{\Delta}_j^+ &= 2\nabla\mathbf{U}_j\cdot\left(\mathbf{x}_j-\mathbf{x}_i\right)-\left(\mathbf{U}_j-\mathbf{U}_i\right)\end{aligned}$$

and the values S_i and S_j are limiters to avoid oscillations next to the shock waves. The van Albada limiter is used:

$$\begin{aligned}S_i &= max\left[0,\frac{2\boldsymbol{\Delta}_i^-\left(\mathbf{U}_j-\mathbf{U}_i\right)+\delta}{\left(\boldsymbol{\Delta}_i^-\right)^2+\left(\mathbf{U}_j-\mathbf{U}_i\right)^2+\delta}\right]\\ S_j &= max\left[0,\frac{2\boldsymbol{\Delta}_j^+\left(\mathbf{U}_j-\mathbf{U}_i\right)+\delta}{\left(\boldsymbol{\Delta}_j^+\right)^2+\left(\mathbf{U}_j-\mathbf{U}_i\right)^2+\delta}\right]\end{aligned}$$

where δ is a small value to prevent the division by zero in smooth zones of the fluid variables.

2.2.2. Evaluation of the viscous terms

Evaluating the viscous terms requires the computation of the gradient of the conservative variables, which is done using the same edge loop of Eq. (2.7):

$$\nabla\mathbf{U}_i=\frac{1}{\vartheta_i}\sum_{j=1}^{\#ed_i}\frac{1}{2}\left(\mathbf{U}_i+\mathbf{U}_j\right)\mathbf{n}_{ij}\sigma_{ij} \tag{2.13}$$

The viscous fluxes are evaluated making use of the gradients, and hence the value of the gradient in the area where the flux is evaluated must be obtained. The simplest approach is taking the

average value between the two points of the edge:

$$\overline{\nabla \mathbf{U}}_{ij} = \frac{1}{2}\left(\nabla \mathbf{U}_i + \nabla \mathbf{U}_j\right), \tag{2.14}$$

but this solution does not prevent the odd-even decoupling of error modes. This is avoided by the numerical diffusion in zones where the viscous terms are not dominant, but in the boundary layer zone that contribution is not enough to prevent the appearance of high frequency modes. The gradient in the edge direction is then substituted by the difference of the variables along the edge [63]:

$$\nabla \mathbf{U}|_{ij} = \overline{\nabla \mathbf{U}}_{ij} - \left(\overline{\nabla \mathbf{U}}_{ij} \cdot \mathbf{l}_{ij} - \frac{\mathbf{U}_j - \mathbf{U}_i}{|\mathbf{x}_j - \mathbf{x}_i|}\right)\mathbf{l}_{ij} \tag{2.15}$$

where $\mathbf{l}_{ij}$ is the unit vector

$$\mathbf{l}_{ij} = \frac{\mathbf{x}_j - \mathbf{x}_i}{|\mathbf{x}_j - \mathbf{x}_i|}$$

The differences between both approaches are presented for a one-dimensional case. In such case, the discrete formulation of the gradient of the conservative variables is

$$\frac{\partial \mathbf{U}_i}{\partial x} = \frac{\mathbf{U}_{i+1} - \mathbf{U}_{i-1}}{2\Delta x}$$

Introducing it in Eq. (2.14) we obtain an expression for the second derivative:

$$\frac{\partial^2 \mathbf{U}_i}{\partial x^2} = \frac{\mathbf{U}_{i+2} - 2\mathbf{U}_i + \mathbf{U}_{i-2}}{4\Delta x^2}$$

This formulation does not prevent the high frequency modes, since abrupt changes between neighbour nodes are not well represented. The modified edge gradient of Eq. (2.15) yields a second derivative:

$$\frac{\partial^2 \mathbf{U}_i}{\partial x^2} = \frac{\mathbf{U}_{i+1} - 2\mathbf{U}_i + \mathbf{U}_{i-1}}{\Delta x^2}$$

which adequately smoothes differences between neighbour nodes.

2.3. Boundary Conditions

Four kinds of boundary conditions are implemented in the code: solid wall, far field, inlet/outlet and phase-lagged periodicity.

The solid wall boundary conditions are imposed making use of the volumes associated to the

wall nodes. For viscous cases with a fixed temperature at the wall, only the mass equation is marched in time imposing that the mass flux at the solid wall is null. If the heat flux is imposed then the energy equation is marched in time making use of this condition. In any case the momentum equation is not used since it is substituted by the condition that the velocity vanishes (or is known) at the wall. For inviscid cases all the equations are marched in time at the solid walls by imposing that the net flux of mass and energy is null and specifying that the contribution of the wall to the momentum equation is only due to the static pressure.

For external flows, the far field boundary condition is imposed based on the Riemann invariants associated to the incoming and outcoming characteristics.

For internal flows, the characteristics theory is used to specify the boundary conditions. For steady cases the radial distribution of stagnation pressure, stagnation temperature and flow angles are imposed at the inlet while the static pressure is specified at the exit. This may be done in a point-wise basis (1D boundary conditions) or as a mean (2D exact steady, non-reflecting boundary conditions [32, 16]). The rest of information is computed making use of the semi-volumes associated to the nodes. 1D and 2D unsteady non-reflecting boundary conditions are also available.

For a cascade vibrating in a travelling-wave mode or a rotor/stator configuration with different number of vanes and blades, phase-lagged boundary conditions are applied in the steady periodic boundaries to perform unsteady computations using a single passage [12].

2.4. Turbulence Model

Turbulence effects are modelled using either the $k-\omega$ two-equation turbulence model [86] or the algebraic Baldwin-Lomax model [7]. When the $k-\omega$ turbulence model is used the conservation equation for the kinetic energy, k, and the dissipation rate, ω, are solved marching in time almost in the same way than the five Reynolds averaged Navier-Stokes equations. The update of the $k-\omega$ equations within each stage of the Runge-Kutta scheme is modified to treat part of the source terms implicitly [52] and increase the numerical stability of the scheme.

2.5. Temporal Discretization

The spatially discretized equations (2.7) can be expressed in the form

$$\frac{d\left(\mathbf{U}_i \cdot \vartheta_i\right)}{dt} = \mathbf{R}\left(\mathbf{U}_i\right) = \mathbf{C}\left(\mathbf{U}_i\right) + \mathbf{D}\left(\mathbf{U}_i\right) \tag{2.16}$$

where the residual has been split in the convective part, $\mathbf{C}$, and in the viscous and numerical diffusion terms, $\mathbf{D}$. This set of coupled ordinary differential equations must be integrated in time to obtain the steady state solution. The basic time-stepping scheme is an explicit five-stage Runge-Kutta scheme [55], where the artificial viscosity and the viscous terms are evaluated only in three stages.

Local time stepping, which is also presented in detail in chapter 5, is used to enhance the convergence acceleration, which guarantees that disturbances will reach the inlet and outlet boundaries in a fixed number of steps proportional to the number of cells between the inner boundary, typically the airfoil, and the outer boundaries.

The residuals are implicitly smoothed by a Jacobi iterative scheme in order to increase the support of the space discretization as well [18, 37]. Only two iterations are employed since the scheme requires a loop over edges, which is an expensive operation for an unstructured solver. This is enough to double the stability limit of the base scheme. Low Mach number preconditioning is also used to accelerate convergence to steady state.

3. Multigrid Algorithm

In this chapter, the fundamentals of the multigrid technique are presented. This technique has been widely used to enhance the convergence rate of classical iterative methods [69, 41, 46, 56], that usually undergo a differential convergence of the error modes. High frequency errors, i.e., errors whose wavelength involves less than four grid points, are appropriately damped, turning the error a smooth function in just a few iterations. On the other hand, low frequency errors have much lower damping factors, and its elimination requires a much larger number of iterations. The multigrid effectiveness relies on the ability of the method to damp these low frequency errors, making use of several grids, each one with a different mesh spacing. When used in conjunction with the multigrid technique, an iterative method is capable of solving a system of equations

$$R_h\left(\mathbf{U}_h, \varphi\right) = 0 \tag{3.1}$$

with N unknowns in $\mathcal{O}\left(N\right)$ operations [10]. In the previous equation, h stands for the mesh spacing and φ is a parameter.

The ideas behind the multigrid algorithm are first presented using a classical model example: the resolution of the one-dimensional Poisson equation by means of a weighted Jacobi iterative method [11]. The model is simple enough to present the drawbacks of the weighted Jacobi method and the contribution of the multigrid algorithm to overcome them. Then, we outline the basic principles of the resolution of hyperbolic problems with the multigrid method, since they are substantially different from those of the elliptic problems. Finally, some practical ideas concerning the implementation of the multigrid algorithm in the baseline solver are outlined, among them, the data storage of the coarser grids, the modifications of the algorithm in the boundaries and the operators required by the multigrid method.

3.1. Resolution of the one-dimensional Poisson equation

Consider the one-dimensional Poisson equation:

$$\begin{aligned} u''(x) &= f(x), \quad x \in [0,1] \\ u(0) &= u(1) = 0 \end{aligned} \tag{3.2}$$

We are solving this problem in a uniform grid of N points using a standard second order discretisation for the second derivative, that leads to the following system of discrete equations:

$$\begin{aligned} u_{j-1} - 2u_j + u_{j+1} &= f_j, \quad 1 \leq j \leq N-1 \\ u_0 &= u_N = 0 \end{aligned} \tag{3.3}$$

The matrix associated with this system of equations is:

$$A_h = \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & -2 \end{bmatrix} \tag{3.4}$$

which is a $N-1 \times N-1$ square matrix. Once formulated the linear system of equations, $A_h \mathbf{u}_h = \mathbf{f}_h$, we may solve it using the iterative weighted Jacobi method:

$$\mathbf{u}_h^{(n+1)} = \mathbf{u}_h^{(n)} - \frac{\omega}{2}\left(A\mathbf{u}_h^{(n)} - \mathbf{f}_h\right) \tag{3.5}$$

In this expression, ω is the relaxation factor, that controls the stability of the iterative scheme.

The error of the solution after n steps is $\mathbf{e}_h^{(n)} = \mathbf{u}_h - \mathbf{u}_h^{(n)}$, which combined with $A_h \mathbf{u}_h = \mathbf{f}_h$ yields

$$A_h \mathbf{e}_h^{(n)} = \mathbf{r}_h \tag{3.6}$$

being $\mathbf{r}_h = \mathbf{f}_h - A\mathbf{u}_h^{(n)}$ the residual. Combining this definition with Eq. (3.5), we obtain the expression of the iterative method for the error:

$$\mathbf{e}_h^{(n+1)} = \left[I - \frac{\omega}{2}A\right]\mathbf{e}_h^{(n)} \tag{3.7}$$

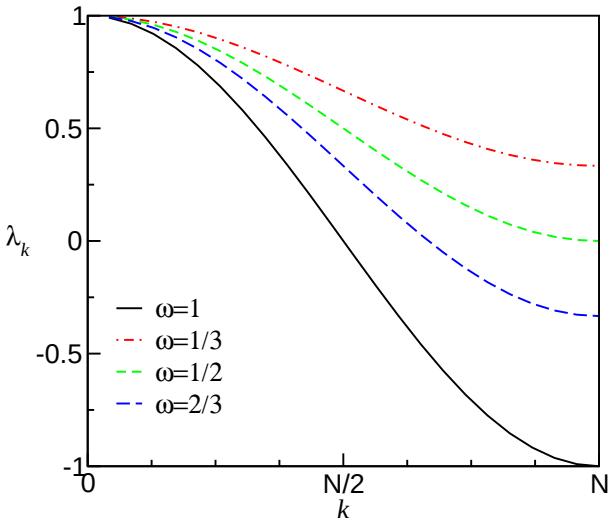


Figure 3.1: Eigenvalues of the operator $\left[I - \frac{\omega}{2}A\right]$ for different wave numbers k.

which recursively applied yields:

$$\mathbf{e}_h^{(n+1)} = \left[I - \frac{\omega}{2}A\right]^{n+1} \mathbf{e}_h^{(0)} \tag{3.8}$$

The eigenvalues and eigenvectors of $\left[I - \frac{\omega}{2}A\right]$ are

$$\lambda_k = 1 - 2\omega \sin^2\left(\frac{k\pi}{2N}\right), \quad k \in [1, N-1] \tag{3.9}$$

and

$$w_{k,j} = \sin\left(\frac{jk\pi}{N}\right), \quad 1 \leq k \leq N-1, \quad 0 \leq j \leq N \tag{3.10}$$

respectively. Expanding the error in the base of eigenvectors we obtain:

$$\mathbf{e}_h^{(0)} = \sum_{k=1}^{N-1} c_k \mathbf{w}_k \tag{3.11}$$

and then Eq. (3.8) can be expressed as

$$\mathbf{e}_h^{(n+1)} = \sum_{k=1}^{N-1} \lambda_k^{n+1} c_k \mathbf{w}_k \tag{3.12}$$

where we have made use of the relation $\left[I - \frac{\omega}{2}A\right]\mathbf{w}_k = \lambda_k \mathbf{w}_k$. Equation (3.12) provides the damping associated with every eigenvector, $\mathbf{w}_k$, after $n+1$ iterations. Two main results are derived from this expression:

1. All λ_k must verify that $|\lambda_k| < 1$ to obtain a convergent scheme. This implies that the

relaxation factor ω must lie between 0 and 1.

2. The largest wavelength of our computational domain has an associated eigenvalue

$$\lambda_1 = 1 - 2\omega \sin^2 \frac{\pi}{2N} = 1 - 2\omega \sin^2 \frac{\pi h}{2} \approx 1 - \frac{\omega \pi^2 h^2}{2} \tag{3.13}$$

The eigenvalues of the system provide the damping of the error modes. Eq. (3.13) shows that the largest wavelength is hardly damped when the grid spacing is small. Moreover, the damping diminishes when the grid is refined. If only this mode were considered, Eq. (3.12) could be expressed as

$$\mathbf{e}_h^{(n+1)} = \lambda_1^{n+1} \mathbf{e}_h^{(0)}$$

or, in logarithmic scale,

$$\ln \mathbf{e}_h^{(n+1)} = (n+1) \ln \lambda_1 + \ln \mathbf{e}_h^{(0)}$$

The slope of the convergence history is $\ln \lambda_1$. Using Eq. (3.13) we obtain that

$$\ln \lambda_1 = \ln \left(1 - \frac{\omega \pi^2 h^2}{2} \right) \approx -\frac{\omega \pi^2 h^2}{2} \tag{3.14}$$

Therefore, the convergence rate of the system is proportional to h^2 when only smooth error modes exist. Figure 3.1 shows the damping for different wave numbers, and also for different values of the Jacobi relaxation factor. For a value $\omega = 2/3$, the oscillatory error waves, i.e.: those whose wavenumber $k > N/2$, are efficiently damped, since their associated eigenvalues are smaller than $1/3$. The eigenvalues of the modes with the largest wavelenghts are close to 1, whatever the value of ω.

Taking into account the damping behaviours explained above, the expected convergence history of the iterative method is that of figure 3.2. The initial iterations show a large reduction of the error, corresponding to the damping of the oscillatory errors. After that, the convergence rate decays, because the oscillatory errors have already been damped and only the smooth errors, which are hard to damp, remain.

One way to increase the damping of a smooth error mode is by coarsening the grid. Let us consider an error function consisting of two harmonics $n_1 < N_h/2$ and $n_2 > N_h/2$, discretised

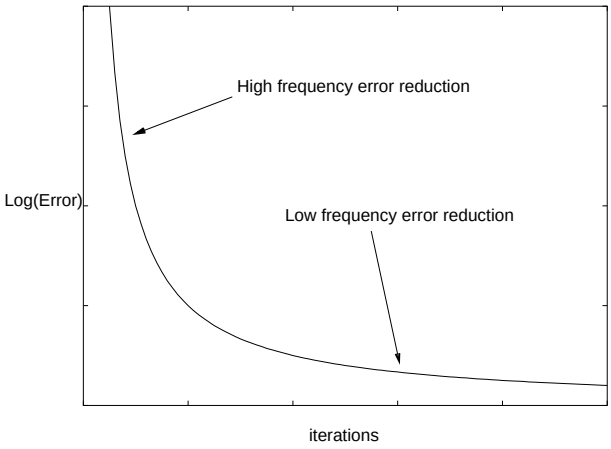


Figure 3.2: Typical convergence history of an iterative method.

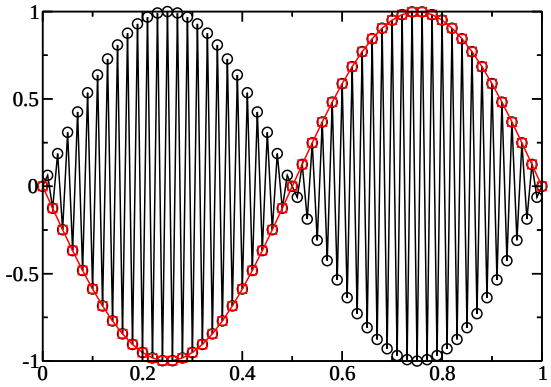


Figure 3.3: Aliasing. $\sin(98\pi x)$, $x \in [0,1]$, **evaluated in 100 equispaced points (black) and in 50 equispaced points (red).**

in a grid of N_h points:

$$e_{j,h}^{(0)} = c_1 \sin\left(\frac{jn_1\pi}{N_h}\right) + c_2 \sin\left(\frac{jn_2\pi}{N_h}\right), \quad 0 \leq j \leq N_h \tag{3.15}$$

The former corresponds to a low frequency mode while the latter is a high frequency one. If we represent this error function in a mesh which has a spacing $H = 2h$, then $N_H = N_h/2$ and the error function of Eq. (3.15) becomes:

$$e_{j,H}^{(0)} = c_1 \sin\left(\frac{jn_1\pi}{N_H}\right) + c_2 \sin\left(\frac{jn_2\pi}{N_H}\right), \quad 0 \leq j \leq N_H$$

It is important to highlight two important points:

1. The first term has an associated damping in the fine grid

$$\lambda_{n_1,h} = 1 - 2\omega \sin^2\left(\frac{\pi n_1}{2N_h}\right) \approx 1 - \frac{\omega\pi^2 n_1^2}{2N_h^2}$$

when $n_1/N_h \ll 1$. In the coarse grid, it turns out to be

$$\lambda_{n_1,H} \approx 1 - \frac{2\omega\pi^2 n_1^2}{N_h^2}$$

that is smaller than $\lambda_{n_1,h}$. This means that the damping of the mode $\lambda_{n_1,H}$ is equal to that of the mode $k_h = 2n_1$ of the fine grid, which is larger. The smooth modes k_h that have an associated damping $\lambda_{k,h}$ in the fine grid have an associated damping $\lambda_{k,H} = \lambda_{2k,h}$ when transferred to the coarse grid.

2. The second term has a somewhat different behaviour. This mode can not be appropriately represented in the coarse grid, since $N_H < n_2$. Hence, it becomes a low frequency mode when projected onto the coarse grid, in a phenomenon known as aliasing. The aliased mode is $\sin\left(j\left(N_h - n_2\right)\pi/N_H\right)$ (see figure 3.3). The aliasing has two undesired consequences: the error function of Eq. (3.15) is misrepresented in the coarse grid, and the damping of the projected mode is given by the eigenvalue $\lambda_{N_h-n_2,H}$, which is higher than that of the fine grid high frequency wave. Thus, the effect of projecting the mode is opposite to the expected.

3.2. Two-level multigrid correction scheme

In the previous section we have described a method to accelerate the convergence of the smooth error waves, by projecting the error function onto a coarser grid. This procedure can be recursively applied to solve the linear system of equations (3.3), yielding the following resolution algorithm, known as the multigrid correction scheme.

1. Smooth ν_1 times $A_h\mathbf{u}_h = \mathbf{f}_h$ on a fine grid until we obtain a smooth approximation $\mathbf{u}_h^{(\nu_1)}$.
2. Compute the residual $\mathbf{r}_h^{(\nu_1)} = \mathbf{f}_h - A\mathbf{u}_h^{(\nu_1)}$.
3. Transfer this residual to a coarse grid to obtain $\mathbf{r}_H^{(\nu_1)} = I_H^h\mathbf{r}_h^{(\nu_1)}$. I_H^h is called the restriction operator, which interpolates residuals from the fine to the coarse grid.
4. Smooth ν_2 times $A_H\mathbf{e}_H = \mathbf{r}_H^{(\nu_1)}$ to obtain an approximate error $\mathbf{e}_H^{(\nu_2)}$.
5. Project this error back again to the fine grid; $\mathbf{e}_h^{(\nu_2)} = I_h^H\mathbf{e}_H^{(\nu_2)}$. I_h^H is the prolongation operator, that assigns the coarse grid error corrections into the fine grid.
6. Correct the solution: $\mathbf{u}_h^{(new)} = \mathbf{u}_h^{(\nu_1)} + \mathbf{e}_h^{(\nu_2)}$.

7. Repeat points 1 through 6 until the convergence has been attained.

When the fine grid problem is exactly solved, the error $\mathbf{e}_h = 0$, then $\mathbf{r}_h = 0$ and the projected error $\mathbf{e}_H$ also vanishes, thus ensuring that no additional corrections are going to be performed and that fine grid solution is the solution of the discrete problem. The algorithm presented here is valid just for two grids and for linear problems, although it can be easily extended to as many grids as desired. The modifications to cope with non-linear problems are explained below.

3.3. Two-level full approximation scheme

When the discrete problem

$$R_h\left(\mathbf{U}_h, \varphi\right) = 0 \tag{3.16}$$

is non-linear, the error equation (3.6) is not valid, hence

$$R_h\left(\mathbf{U}_h, \varphi\right) - R_h\left(\mathbf{U}_h^{(n)}, \varphi\right) \neq R_h\left(\mathbf{e}_h^{(n)}, \varphi\right)$$

The multigrid correction scheme, that was based on the linearity of the problem, is not valid. Now the equation (3.6) for the error after n iterations is

$$R_h\left(\mathbf{U}_h^{(n)} + \mathbf{e}_h^{(n)}, \varphi\right) - R_h\left(\mathbf{U}_h^{(n)}, \varphi\right) = \mathbf{r}_h$$

where the residual $\mathbf{r}_h = -R_h\left(\mathbf{U}_h^{(n)}, \varphi\right)$. After n iterations in the fine grid, the error is supposed to be smooth, therefore we can transfer it to the coarse grid, yielding:

$$R_H\left(I_H^h \mathbf{U}_h^{(n)} + \mathbf{e}_H^{(0)}, \varphi\right) - R_H\left(I_H^h \mathbf{U}_h^{(n)}, \varphi\right) = I_H^h \mathbf{r}_h \tag{3.17}$$

If we introduce a new coarse grid variable

$$\mathbf{U}_H^* = I_H^h \mathbf{U}_h^{(n)} + \mathbf{e}_H^{(0)}, \tag{3.18}$$

Eq. (3.17) is expressed as

$$R_H\left(\mathbf{U}_H^*, \varphi\right) = \mathbf{f}_H,$$

which is a system of equations for the coarse grid which is analogous to that of Eq. (3.16) with a new forcing term

$$\mathbf{f}_H = R_H\left(I_H^h \mathbf{U}_h^{(n)}, \varphi\right) - I_H^h R_h\left(\mathbf{U}_h^{(n)}, \varphi\right) \tag{3.19}$$

The new variable $\mathbf{U}_H^*$ is the sum of the restricted fine grid variable and its correction, that is why this scheme is called full approximation. Once enough iterations are performed in the coarse grid to obtain a smooth solution, the correction may be transferred back to the fine grid, yielding a correction for the solution in the fine grid:

$$\mathbf{U}_h^{(new)} = \mathbf{U}_h^{(n)} + I_h^H \mathbf{e}_H,$$

which can be expressed as

$$\mathbf{U}_h^{(new)} = \mathbf{U}_h^{(n)} + I_h^H\left(\mathbf{U}_H^* - I_H^h \mathbf{U}_h^{(n)}\right), \tag{3.20}$$

making use of Eq. (3.18).

3.4. Cycling strategies

It is common to use more than two grid levels to solve a system of equations, since some of the error modes remain smooth even after transferring them to a coarser grid (those whose wavelength $k < N_h/4$). After performing some iterations in a coarser grid, the error becomes smooth again, and it should be transferred again to a coarser grid to efficiently damp them. This process leads to a nested multigrid algorithm, where cycles of interpolation and restriction appear. Not all of the cycles provide the same robustness and convergence rate. Here we present the most common ones, namely the V-cycle and the W-cycle.

For each of these cycles, the number of smoothing iterations performed at each grid level may be kept constant or variable. The latter approach is more efficient: we perform as many iterations as needed to damp all the high frequency error mores, thus minimising the number of iterations in each grid level. But the former approach is simpler, hence easier to implement, and the CPU cost is not severely increased.

The V and the W multigrid cycles are shown in figure 3.4. The use of W-Cycle is often considered more efficient and robust. In every grid level we must run enough iterations to ensure that the high frequency errors have been damped. This number will depend on the

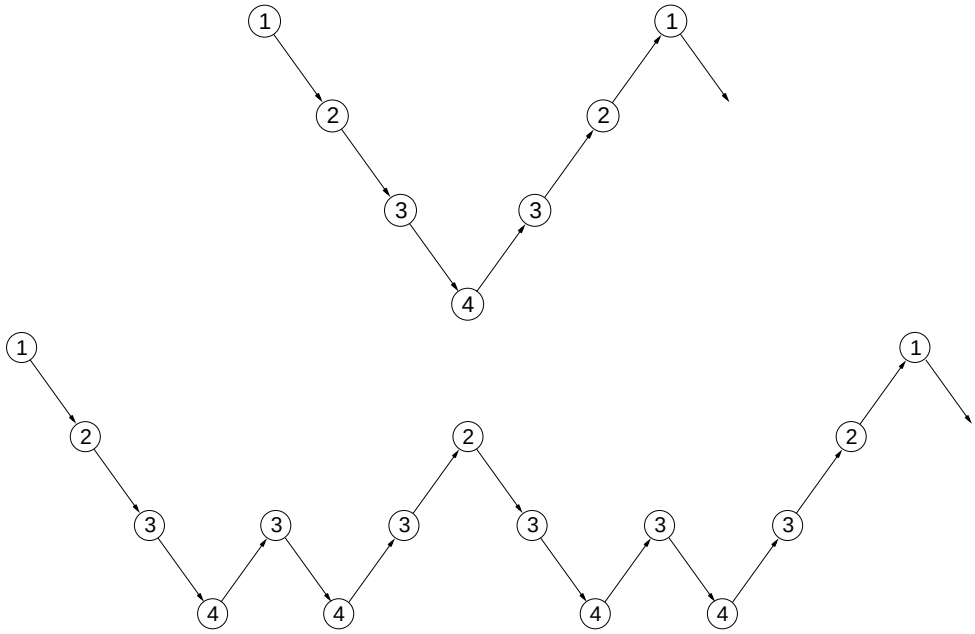


Figure 3.4: Common 4-levels Multigrid Cycles. V-Cycle (top); W-Cycle (bottom).

ability of the iterative method to damp the high frequency error modes of the computational domain. Usually, iterations are run in both ascending and descending paths, but the number of them can vary depending on the sense, running more iterations when descending and less when ascending. When no iterations are run in the ascending paths, we obtain the so called Saw Tooth Cycles, that are commonly used, too.

The cost of each type of cycle for a two-dimensional problem, assuming a coarsening ratio of 4:1 and the same number of iterations in each level is, relative to that of single grid:

V-Cycle

$$1 + 2 \times \frac{1}{4} + 2 \times \frac{1}{16} + 2 \times \frac{1}{64} + \ldots < 1 + 2 \times \frac{1}{4} \sum_{i=0}^{\infty} \frac{1}{4^i} = \frac{5}{3} \tag{3.21}$$

W-Cycle

$$1 + 3 \times \frac{1}{4} + 6 \times \frac{1}{16} + 10 \times \frac{1}{64} + \ldots < 1 + \frac{1}{4} \left(3 + 2 \cdot \sum_{i=1}^{\infty} \frac{2i+1}{4^i} \right) = \frac{85}{36} \tag{3.22}$$

In three dimensions, where the coarsening ratio is 8:1, these factors are 9/7 for V-Cycle and 1.493 for W-Cycle.

It is common, for starter procedures, to establish a ramp to improve the initial solution on the fine grid; this is called the Full Multigrid Strategy (see figure 3.5).

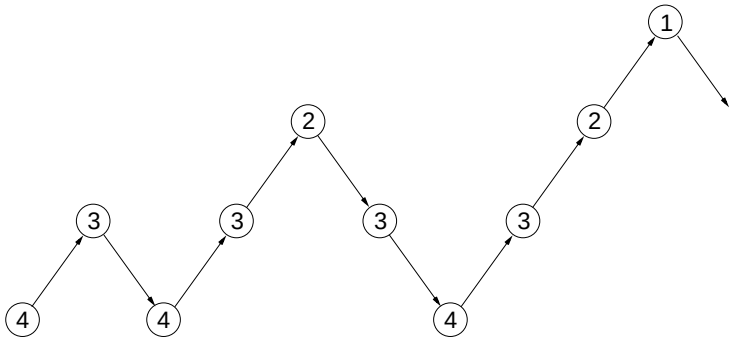


Figure 3.5: Full Multigrid Strategy.

3.5. The multigrid technique in hyperbolic problems

Consider the one-dimensional linear wave equation:

$$\begin{aligned} &\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0, \qquad x \in [0, 2\pi] \\ &t = 0 \Rightarrow u = u\left(x\right), \quad \forall x \in [0, 2\pi] \\ &u(0) = u(2\pi), \qquad \forall t \end{aligned} \tag{3.23}$$

We discretise the spatial derivative with an equispaced grid of N points using a first order upwind scheme, yielding the following set of ordinary differential equations:

$$\begin{aligned} &\frac{du_i}{dt} + c \cdot \frac{u_i - u_{i-1}}{\Delta x} = 0, \quad 0 \leq i \leq N, 1 \leq n \leq nmax \\ &u_i^0 = u_{0,i}, \qquad\qquad 0 \leq i \leq N \\ &u_0 = u_N, \end{aligned} \tag{3.24}$$

If we express the solution as a Fourier series:

$$u = \sum_{k=-N/2}^{k=N/2} \widehat{u}_k\left(t\right) e^{ikx}$$

we may obtain the evolution equation for each wavenumber k:

$$\frac{d\widehat{u}_k}{dt} + ikc\widehat{u}_k = 0$$

The solution to this equation is:

$$\widehat{u}_k\left(x,t\right) = \widehat{u}_{0,k} e^{ik(x-ct)} \tag{3.25}$$

If we introduce the Fourier series in the discrete system of equations (Eq. (3.24)), we obtain

$$\frac{d\widehat{u}_{k,i}}{dt} + \widehat{u}_{k,i} c \cdot \frac{\left(1 - \cos\widehat{k}\right) + i \cdot \sin\widehat{k}}{\Delta x} = 0$$

where $\widehat{k} = k\Delta x$. The solution to this equation is

$$\widehat{u}_{k,i}\left(x_i, t\right) = \widehat{u}_{0,k} e^{-kc\frac{1-\cos\widehat{k}}{\widehat{k}}t} e^{ik\left(x_i - c\frac{\sin\widehat{k}}{\widehat{k}}t\right)} \tag{3.26}$$

When comparing Eqs. (3.25) and (3.26), we can notice some differences introduced by the spatial discretisation:

- The discrete solution has a damping term, $e^{-kc\frac{1-\cos\widehat{k}}{\widehat{k}}t}$, that does not exist in the solution of the linear wave equation.

- The propagation term has also been modified. In the continuous solution, the waves are propagated with a velocity c, while in the discrete approach, these are propagated with a velocity $c \cdot \sin\widehat{k}/\widehat{k}$. By dividing these two numbers we obtain a relation between the numerical and physical propagation speeds, the so called numerical phase velocity:

$$\frac{\widehat{c}}{c} = \frac{\sin\widehat{k}}{\widehat{k}} \tag{3.27}$$

In the solution of Eq. (3.26) we can distinguish two limits:

- For $\widehat{k} = \pi$, that represents the shortest wavelength, Eq. (3.26) reduces to

$$\widehat{u}_{k,i}\left(x_i, t\right) = \widehat{u}_{0,k} e^{-\frac{2kc}{\widehat{k}}t}$$

  This wave is just damped, not propagated. The numerical phase velocity of Eq. (3.27) is $\widehat{c}/c = 0$. For this wave number, the discrete problem behaves as an elliptic equation, hence the strategies for converging this error will be similar to those presented in section 3.1.

- For long wavelengths, or smooth errors ($\widehat{k} \ll 1$)

$$\widehat{u}_{k,i}\left(x_i, t\right) = \widehat{u}_{0,k} e^{ik(x_i - ct)} \tag{3.28}$$

since $\cos\widehat{k} \approx 1$ and $\sin\widehat{k} \approx \widehat{k}$ when $\widehat{k} \ll 1$. In this case the damping is much lower than that of short wavelengths, but now the waves are correctly transported ($\widehat{c}/c \approx 1$).

Therefore, the benefits of the multigrid algorithm for hyperbolic equations are twofold:

1. If we advance in time with an explicit scheme, the maximum Δt is limited due to stability constraints, thus the convection of the errors cannot be as fast as desired. By halving the number of points, the maximum allowed Δt can be doubled and errors are advected twice as faster in the coarse grid.

2. When we transfer the errors into a coarser level, the error waves turn sharper. Certain waves are not convected anymore, according to Eq. (3.26), since their wave number yields a phase velocity $\widehat{c}/c \ll 1$. These waves are damped by the iterative scheme.

3.6. Results

In this section, the system of Eq. (3.3) is solved using the multigrid correction scheme presented in section (3.2), together with a weighted Jacobi iterative method with a value of the weight $\omega = 2/3$. The term $f(x)$ of Eq. (3.3) is set to zero, hence the final solution to the problem is $u_h = 0$. The initial solution is set to:

$$u_h^{(0)} = \sin(2\pi x) + \frac{1}{2}\sin(4\pi x) + \sin(20\pi x)$$

We have solved this problem using two fine grids of 512 and 1024 points. The convergence rate of the problem is dominated by the damping of the low frequency modes of the initial solution ($\sin(2\pi x)$ and $\sin(4\pi x)$), since the high frequency component ($\sin(20\pi x)$) is damped in just a few iterations.

Figure 3.6 compares the convergence histories when the weighted Jacobi is used without the multigrid method. The number of iterations has been non-dimensionalised with the number of iterations needed to reach the solution in the finest grid. We can see that the convergence rate decays as the number of grid points is increased. According to Eq. (3.14), the relation between convergence slopes for two grids $h1$ and $h2$ is $(N_{h1}/N_{h2})^2$ where N_{h1} and N_{h2} are the number of points of these grids.

Figure 3.7 compares the convergence of the cases where the multigrid has been used. For both cases, the convergence rate has been dramatically improved due to the use of the multigrid

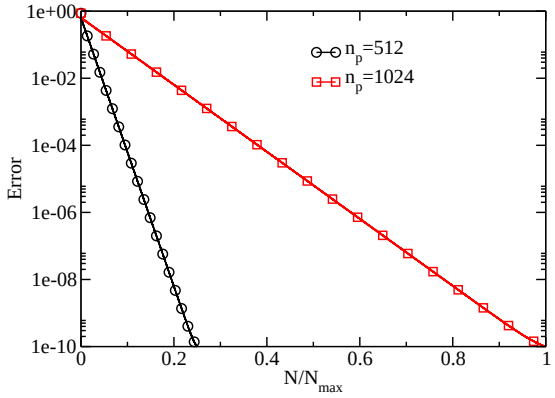


Figure 3.6: Convergence of the Poisson equation using a weighted Jacobi iterative method.

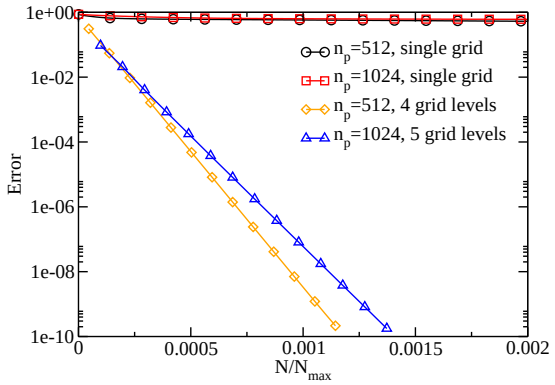


Figure 3.7: Comparison of the convergence of the Poisson equation using multigrid.

method. It is observed that the number of iterations required to converge the system of equations is approximately the same when the multigrid method is used (only 15% more iterations are needed to converge the finer case). This is a characteristic of the multigrid algorithm: the cases are converged with a number of operations $\mathcal{O}(N_p)$, whatever the value of N_p, hence the number of iterations is not substantially modified.

Results for hyperbolic equations will be presented when discussing the application of the multigrid algorithm to solve the Reynolds-Averaged Navier-Stokes equations.

3.7. Implementation of the Multigrid Algorithm

This section explains the solutions adopted to implement the multigrid algorithm in the baseline solver described in chapter 2. Two main difficulties are presented, namely, the integration of all the grids in a single data structure, and the obtention of the multigrid forcing term of Eq. (3.19), which requires a careful treatment of the boundary conditions. On the other hand, the restriction and prolongation operators used to transfer the error between grids must be defined for the unstructured grids used in the baseline code.

3.7.1. Multigrid Data Structure

The minimum amount of data required to perform a single grid simulation are, according to Eq. (2.7):

- The conservative variables at each node $\mathbf{U}_i$.
- The control volume of each node ϑ_i.
- The edges that connect the grid nodes.
- The areas associated to these edges, $\sigma \cdot \mathbf{n}$.

This data structure should be used for all the grids involved in the multigrid algorithm, i.e., we should use a single array to store the unknown variables $\mathbf{U}$ for all the grids, another array for the control volumes, etc. The coarse grid data are stored immediately after the fine grid data. If the conservative variables in a single grid are stored in an array $\mathbf{U}(\texttt{nnode}_{\texttt{fine}})$, when using two grids this array is sized $\mathbf{U}(\texttt{nnode}_{\texttt{fine}} + \texttt{nnode}_{\texttt{coarse}})$, and the conservative variables of the coarse grid are stored between $\mathbf{U}(\texttt{nnode}_{\texttt{fine}} + 1)$ and $\mathbf{U}(\texttt{nnode}_{\texttt{fine}} + \texttt{nnode}_{\texttt{coarse}})$. The situation is analogous for the rest of the variables.

We control the pointer to the first node of each grid and the number of nodes in that grid using an additional array named `nnodeMG`. The steps to construct it are:

1. Initialise $\texttt{NMG} = 1$. `NMG` stands for the grid level. The finest grid level is $\texttt{NMG} = 1$ and the coarsest one, $\texttt{NMG} = \texttt{Number\_Grids}$. Set $\texttt{nnodeMG}(1) = 0$, and $\texttt{nnodeMG}(2) = \texttt{nnode}_{\texttt{fine}}$.
2. For each coarse grid level, i.e., for $\texttt{NMG} = 2$ to $\texttt{NMG} = \texttt{Number\_Grids}$:

$$\texttt{nnodeMG}(\texttt{NMG} + 1) = \texttt{nnodeMG}(\texttt{NMG}) + \texttt{nnode}_{\texttt{coarseNMG}},$$

   where $\texttt{nnode}_{\texttt{coarseNMG}}$ is the number of nodes of the grid level `NMG`.

These steps are performed in parallel with the grid construction, which is addressed in chapter 4.

Therefore, when we are smoothing the errors in the grid level `NMG`, we know that the first element of the array that belongs to that grid is $\texttt{nnodeMG}(\texttt{NMG}) + 1$, and that the number of nodes of the grid is $\texttt{nnodeMG}(\texttt{NMG} + 1) - \texttt{nnodeMG}(\texttt{NMG})$. We must create another array analogous to

`nnodeMG` for the number of edges and in general for every grid element we use (boundary edges, etc.).

With this approach, only the routine calls are modified. The routines themselves may be placed in separate libraries and we don't need to modify them. If the routine call used in a single grid code is:

$$\texttt{EvaluateFlux}(\texttt{U}, \texttt{nnode}, ...)$$

in a multigrid environment would be:

$$\texttt{EvaluateFlux}(\texttt{U}(\texttt{nnodeMG}(\texttt{NMG}) + 1), \texttt{nnodeMG}(\texttt{NMG} + 1) - \texttt{nnodeMG}(\texttt{NMG}), ...)$$

Depending on the value of `NMG`, we may work with one grid level or another.

3.7.2. Computation of the forcing term

The forcing term of Eq. (3.19) must ensure that when the problem is converged, no additional correction is performed due to the use of the multigrid algorithm. The system of equations is fully converged when Eq. (3.16) is fulfilled. Introducing this result into Eq. (3.19) yields the expression of the multigrid forcing function for a converged problem:

$$\mathbf{f}_H^{converged} = R_H \left(I_H^h \mathbf{U}_h^{(n)}, \varphi \right)$$

When we compute the residuals in the coarse grid, $\mathbf{F}_H$, we obtain:

$$\mathbf{F}_H = R_H \left(I_H^h \mathbf{U}_h^{(n)}, \varphi \right) - \mathbf{f}_H^{converged},$$

which yields $\mathbf{F}_H = 0$. Thus, when we update the variables

$$\mathbf{U}_H^* = I_H^h \mathbf{U}_h^{(n)} + \mathbf{F}_H,$$

we obtain that $\mathbf{U}_H^* = I_H^h \mathbf{U}_h^{(n)}$. After this step, we should impose the boundary conditions, provoking a modification of the boundary variables. Hence, the fine grid variable $\mathbf{U}_h^{(new)} \neq \mathbf{U}_h^{(n)}$, according to equation (3.20), preventing the convergence of the method. To avoid this situation we must modify the system of equations (3.16) in the boundaries:

1. Compute the fine grid fluxes

$$\mathbf{F}_h = R_h\left(\mathbf{U}_h^{(n)}, \varphi\right)$$

2. Update the variables

$$\mathbf{U}_h^{(n+1)} = \mathbf{U}_h^{(n)} + \mathbf{F}_h$$

3. Impose the boundary conditions, yielding a modified fine grid boundary variables $\mathbf{U}_h^{*(n+1)}$.

4. Compute a new value of the residual in the boundary:

$$R_h^*\left(\mathbf{U}_h^{(n)}, \varphi\right) = \mathbf{U}_h^{*(n+1)} - \mathbf{U}_h^{(n)}$$

The forcing function of Eq. (3.19) is computed with the modified residual instead of the original one. With this change, the coarse grid variables are not modified after performing one iteration, neither the fine grid ones, and the multigrid algorithm converges to the steady solution.

3.7.3. Inter-grid Transfer Operators for Unstructured meshes

The coarsening strategy that has been implemented is based on an agglomeration method. This approach, that will be presented in the next chapter, consists in fusing fine grid neighbour control volumes to create an agglomerated coarse grid control volume. The inter-grid transfer operators are quite simple when treating such nested volumes. The restriction operator I_H^h, that transfers the fine grid residuals to the coarse grid, is a volume-weighted average:

$$\mathbf{U}_{k,H} = \frac{1}{\vartheta_{k,H}} \sum_{i=1}^{\vartheta_{i,h}\in\vartheta_{k,H}} \vartheta_{i,h} \cdot \mathbf{U}_{i,h} \tag{3.29}$$

where ϑ is the control volume. The prolongation operator I_h^H is just an injection of the coarse grid variables onto the fine grid

$$\mathbf{U}_{i,h} = \mathbf{U}_{k,H}, \qquad \forall \vartheta_{i,h} \in \vartheta_{k,H}, \tag{3.30}$$

4. Coarsening Strategy in Unstructured Grids

Multigrid methods require a procedure for building coarser grids. This method may consist in either the manual generation of as many grids as desired or the automated generation of coarser grids departing from an initial fine grid. This second approach seems more attractive from the user's point of view, since focus can be exclusively placed in constructing a suitable fine grid. Thus, a method for building coarser meshes must be developed.

This method may be very simple in structured grids (e.g.: remove one out of two points in every grid direction), but requires more elaborated techniques when dealing with unstructured meshes. In that sense several techniques are at hand, among them the direct agglomeration, and the edge-collapsing methods. Another approach consists in inverting the sequence and starting from a coarse mesh that is recursively subdivided to obtain the finer grid. This method has some advantages when designing the restriction and prolongation operators since the exact way of grid generation is known (usually additional nodes are positioned in the mid-position of the existing edges). This approach strongly couples the multigrid strategy with the mesh construction. However, to build the initial mesh, the designer should take into account its impact on the finest mesh. The convergence rate of the method is determined by the coarsest grid, then a very coarse mesh is desired in the last grid level. However, a very coarse initial mesh leads to poor quality fine grids, that may spoil the accuracy of the method. Another approach consists in constructing as many meshes as desired grid levels [56], each one coarser than the previous by a factor of 4 in two-dimensional grids and 8 in three-dimensional grids. This approach is not convenient from the user's point of view because the user should exclusively focus on constructing a suitable fine grid. The manual generation of coarser meshes is an additional burden for the designer, since the additional grids are only used to accelerate the convergence of the algorithm and are supposed not to affect the final solution.

Therefore, an automated way of generating coarse meshes is always preferred. Two main approaches are available when dealing with unstructured grids, namely the edge-collapsing and the volume agglomeration methods. The former consists in the removal of fine grid edges to

conform a coarser grid with less elements [66, 65, 85]. The advantage of such method, as previously pointed out, is that the grid quality is controlled in the coarser levels, thanks to geometric restrictions imposed when generating the coarser elements. However, the use of too many restrictions limits the generation of new elements, leading to poor coarsening ratios. The geometric restrictions must be robust enough to prevent the generation of degenerated cells in the coarser meshes, but still have to allow acceptable coarsening ratios. This equilibrium is not always obvious. The direct volume agglomeration approach [57, 58, 60] inverts the advantages and inconvenients of the edge collapsing method. Since the coarser elements are constructed agglomerating finer cells, the volumes of the new elements are always positive by construction, without the need to resort to any geometric constraints. The lack of constraints in the coarsening process leads to poor quality grids. This is especially undesirable in high-gradient regions such as boundary layers, where discretisation errors may affect the solution very much. The latter approach has been chosen, since it provides a greater flexibility when generating the coarse mesh, and it is also easier to implement in an edge-based algorithm, without need to explicitly take into account different types of elements. It is also better suited for a further parallelisation of the code, since the coarser meshes are nested and the different sub-domains resulting from the partition of the original fluid domain can be agglomerated independently. This property simplifies the communication strategy of the parallel algorithm, since fine grids can transfer the solution to coarser ones without passing information from one domain in the fine grid to another one in the coarser, and vice versa; a correction of the coarser mesh in one domain is passed to the same domain of the fine grid, without any inter-domain communication [19].

The method we are using is control volume based and consists in fusing neighbouring control volumes to form an agglomerated volume. This approach works in both cell-centered and vertex-based schemes. We will pay special attention to the second case, since that is used in the baseline solver. The agglomeration of the control volumes leads to polygonal elements in the coarse level. The fact that coarser levels do not preserve the original element shape makes impossible the use of any particular element definition when agglomerating the grid. The only topological structures preserved when agglomerating are the nodes (and the control volume associated to them) and the edges. Hence, the main requirement of the agglomeration algorithm is that only the nodes and edges of the fine grid must be used to construct a coarser grid, that consists in agglomerated volumes and coarse grid edges. The resulting algorithm is only well-suited for edge-based data structure solvers.

We will also explain in detail some enhancements of the baseline coarsening algorithm to deal with quad elements, hexahedra, highly stretched quad cells and three-dimensional semi-unstructured meshes. All these cases have one thing in common: not all the nodes of the element are directly connected by an edge. This characteristic makes the basic agglomeration algorithm unsuitable, because the edge-based agglomeration of neighbouring volumes leads to very irregular geometries. Fixing these inconvenients improves both the coarsening ratios and the quality of the coarse grids.

4.1. Building edge-based data structure for coarser grids

The starting point of the agglomeration algorithm is a set of control volumes $\vartheta_{i,h}$ and the connectivity among them. A list of edges, called `ned`, that has two elements per edge, stores the indexes of the nodes that define every edge. This array has dimensions `ned(2,number_edges)`. Every edge has an associated area $\sigma_{ij}\mathbf{n}_{ij}$ that is used to compute the fluxes using Eq. (2.7).

An auxiliary array, called `edgnod`, is required to construct the volumes and edges of the coarse grid. This array contains, for each node, the number of edges that touch it, and also its indexes. Therefore, this array has dimensions `edgnod(1+max(edges_per_node),number_nodes)`, where `max(edges_per_node)` is the maximum number of edges that reach a node of the grid. It is generated in a pre-processing step in the following way:

```
do i=1,nedge
  i1=ned(1,i)
  i2=ned(2,i)
  edgnod(1,i1)=edgnod(1,i1)+1
  edgnod(1,i2)=edgnod(1,i2)+1
  edgnod(1+edgnod(1,i1),i1)=i
  edgnod(1+edgnod(1,i2),i2)=i
enddo
```

Once `edgnod` has been constructed, we can start the edge-based agglomeration process. The basic agglomeration algorithm [57, 60] proceeds as follows:

1. Choose a seed point. It is coloured in dark blue in figure 4.1b. The seed point order of choice strongly determines the overall agglomeration ratio, therefore a seed point choosing

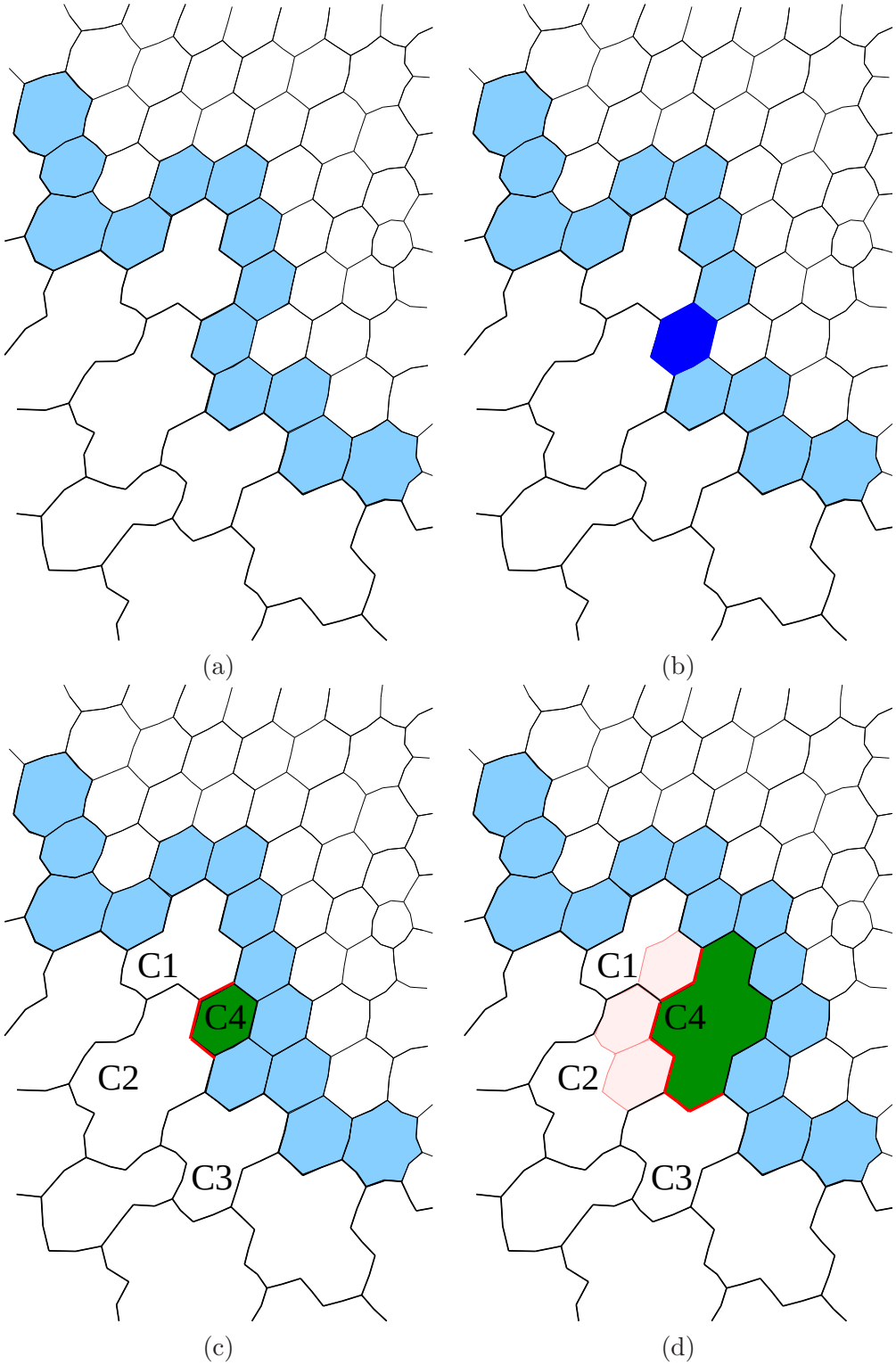


Figure 4.1: Single coarse grid node creation. (a) A front of seed node candidates, coloured in light blue, separates the agglomerated and non-agglomerated control volumes. (b) A seed point, coloured in dark blue, is selected from the front. (c) The agglomeration sequence is started, generating a new coarse grid node C4, and two new edges C1-C4 and C2-C4. (d) After looping over the edges that reach the seed node, we have agglomerated the available ones, extending the coarse grid node C4. The edges C1-C4 and C2-C4 are updated and the edge C3-C4 is created. The front of seed node candidates is updated every time a node is agglomerated.

algorithm has been implemented. An agglomeration front of points is generated, which is initially empty, since we have not agglomerated any point yet. The points of the front are those which have at least one neighbour point agglomerated. Using the front of seed point candidates, we establish the following order of seed point choice:

a) A point on the front and on the inner boundaries.

b) A point on the inner boundaries but not on the front.

c) A point on the front and on the outer boundaries.

d) A point on the outer boundaries but not on the front.

e) The first non-agglomerated point found.

The front of seed point candidates looks like the one depicted in figure 4.1a, where the dual mesh that is being agglomerated is presented. It may be appreciated how the coarse and fine grid control volumes are separated by the front nodes, whose control volumes have been coloured in light blue.

2. Agglomerate the seed point to create a new coarse grid node. Performing this agglomeration involves the obtention of necessary data to carry out the simulation in the coarser meshes, that is, coarse grid control volumes, edges and edge associated areas. It is obtained by executing a sequence of operations each time we agglomerate a fine grid node. This sequence will be referred to as the basic agglomeration step and it constitutes the kernel of the agglomeration process:

   a) Create a new coarse grid node if it has not been created yet, whose initial value of the control volume is that of the fine grid node:

$$\vartheta_j|_{\texttt{coarse\_grid\_node}} = \vartheta_i|_{\texttt{fine\_grid\_node}}$$

   If it has been created, then add the control volume of the fine grid node to the agglomerated coarse grid node:

$$\vartheta_j|_{\texttt{coarse\_grid\_node}} = \vartheta_j|_{\texttt{coarse\_grid\_node}} + \vartheta_j|_{\texttt{fine\_grid\_node}}$$

b) Loop over the edges that reach the fine grid node. The number of edges is:

$$\texttt{edges\_per\_node} = \texttt{edgnod}(1, \texttt{fine\_grid\_node})$$

according to the array description given in section 4.1. For each edge `neighbour_edge`:

i. Choose the neighbour of the fine grid node: `neighbour_node`.

ii. If `neighbour_node` has not been previously agglomerated, put it in the front.

iii. If `neighbour_node` has already been agglomerated but it is not in the same agglomerated node `coarse_grid_node` of `fine_grid_node`, then `neighbour_edge` belongs to a new frontier that separates two coarse grid nodes. This edge could have been previously created or not. Data for this coarse grid edge must be updated in any case.

A. If the edge has been previously created, just the edge associated area must be updated:

$$\sigma_{kl}\mathbf{n}_{kl}|_{\texttt{coarse\_edge}} = \sigma_{kl}\mathbf{n}_{kl}|_{\texttt{coarse\_edge}} + \sigma_{ij}\mathbf{n}_{ij}|_{\texttt{neighbor\_edge}}$$

where the indexes i and j stand for the fine grid nodes that define `neighbour_edge`.

B. If not, increase by one the number of new edges and create both the `ned` array entry and its associated area for the new edge.

$$\begin{aligned} \texttt{ned}(1, \texttt{new\_coarse\_grid\_edge}) &= \texttt{coarse\_grid\_node}_k \\ \texttt{ned}(2, \texttt{new\_coarse\_grid\_edge}) &= \texttt{coarse\_grid\_node}_l \\ \sigma_{kl}\mathbf{n}_{kl}|_{\texttt{coarse\_edge}} &= \sigma_{ij}\mathbf{n}_{ij}|_{\texttt{neighbor\_edge}} \end{aligned}$$

After performing the basic agglomeration step the status of the agglomeration is that of figure 4.1c. The new coarse grid node C4 has been created (it has been coloured in green) and has a volume that, at the moment, is equal to that of the fine grid seed node. A new fine grid volume has been added to the front of seed node candidates, and two new coarse grid edges and associated areas have been created, one connecting nodes C1 and C4, and the other connecting nodes C2 and C4.

3. Loop over the edges that reach the seed node and agglomerate the available fine grid nodes. Every time a fine grid node is agglomerated, the step 2 is repeated. The result of this

step is seen in figure 4.1d. Three new fine grid volumes have been added to the coarse grid node C4 (all of them are coloured in green). The rest, coloured in light red, have not been agglomerated since they belong to other coarse grid nodes. As a consequence, the edges that were previously created when agglomerating the seed point are updated, and an additional edge joining coarse grid nodes C3 and C4 is added to the edge list. The front of seed node candidates is also updated.

The result of applying the basic agglomeration algorithm to the dual grid (figure 4.2b) of an initial grid made up of triangles (figure 4.2a) are the coarse grids shown in figures 4.2c and 4.2d.

Another phenomenon observed in figure 4.1 is that seemingly, the agglomeration ratio obtained for nodes is not preserved for edges. However, the generation of the coarse grid following the algorithm previously described avoids having concatenated edges, that is, a set of edges that connects the same pair of nodes, replacing the complex coarse grid volume frontiers by a straight line (see figure 4.3). This solution does not suppose any approximation in the coarse grid, since the variables in each node of the edge are the same for all the frontier edges, and the flux integration along this frontier yields the same result over the segmented as over the straight edge.

The cost of the whole coarsening process is proportional to the number of fine grid nodes, and represents a very small fraction of the time spent on the convergence to the steady state.

4.2. Agglomerating quad elements

Quad elements represent a problem for edge-based agglomeration algorithms. When looping over the surrounding edges of a seed node in quad elements, the nodes positioned in the quad diagonal do not see each other. If we would simply agglomerate the neighbouring nodes of the seed node, we would obtain a cross-shaped agglomerated volume (see figure 4.4 left). This degrades the fine to coarse grid edge ratio, to a value much lower than the expected 4:1 ratio in two dimensions.

This problem would vanish if the scenario were that presented in figure 4.4 right. Besides, this agglomeration pattern preserves grid regularity, yielding a structured mesh of quads if we depart from a structured grid. Fixing this problem requires another array analogous to the `edgnod` array that, for each node, provides the quad diagonals. This new array needs to be obtained without recurring to element type information. Therefore, the quad recognition

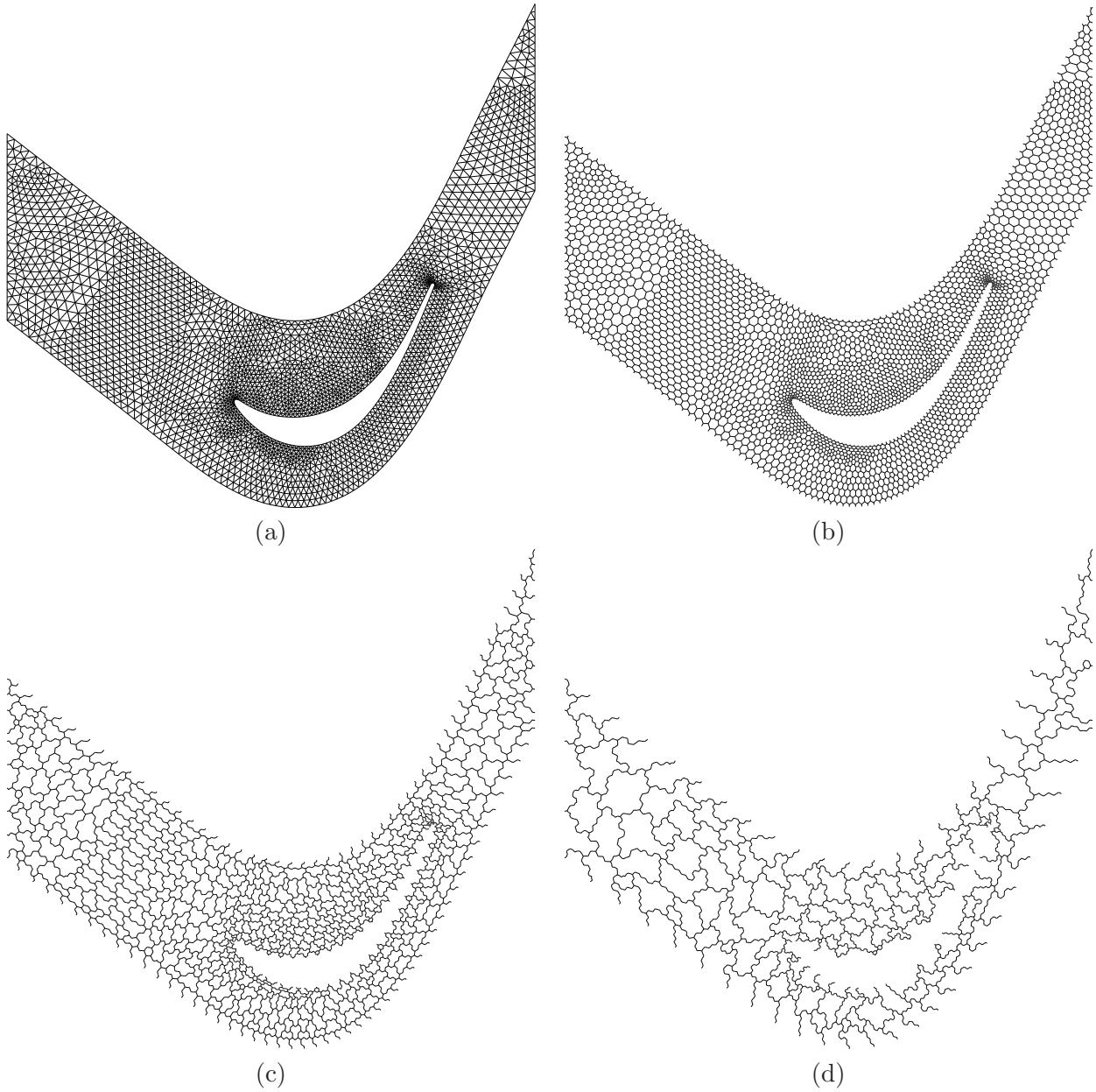


Figure 4.2: Grid agglomeration sequence on a T106 grid. (a) Fine mesh. 2869 nodes; (b) Fine dual mesh; (c) First agglomerated level. 698 nodes; (d) Second agglomerated level. 188 nodes.

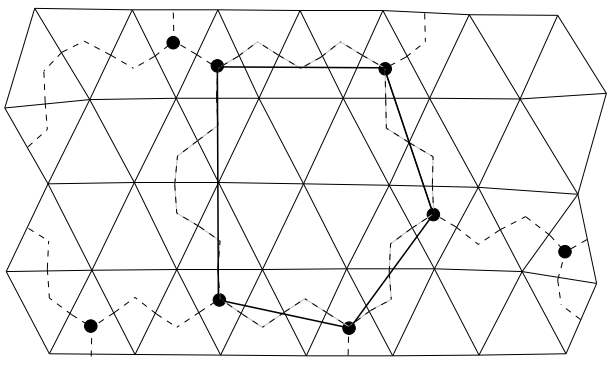


Figure 4.3: The segmented agglomerated edge is replaced by a straight edge in the coarse grid.

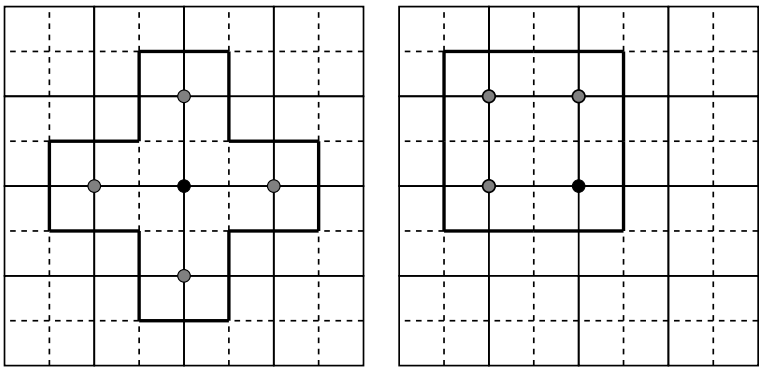


Figure 4.4: The problem of agglomerating quad elements. Left: Nodes in the quad diagonal are not connected by edges, giving rise to this pathological case. Right: The intuitive way of agglomerating quads, preserving grid regularity.

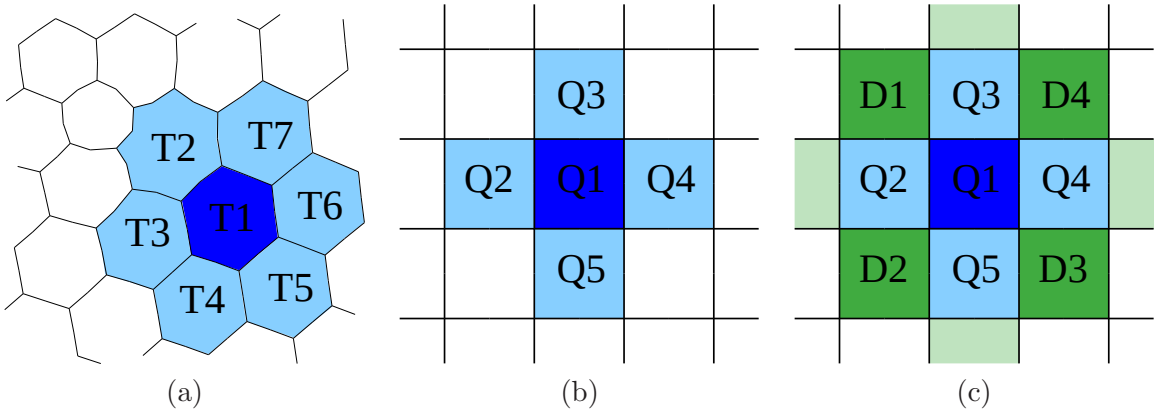


Figure 4.5: Sketch of the algorithm to obtain the diagonal nodes: Initial node (T1 and Q1) and its neighbours (T2 to T7 and Q2 to Q5) are flagged. For each neighbouring node we loop over its edge neighbours. (a) If we found two previously flagged nodes, the element is a triangle. (b) Otherwise it is a quad. (c) In the second case, the diagonal nodes are those which are neighbours of two Qi neighbouring nodes.

algorithm is also edge-based, like the agglomeration basic step. The steps to obtain the diagonal nodes are:

1. Choose the initial node (T1 in figure 4.5a or Q1 in figure 4.5b), and flag its neighbours (T2 to T7 in figure 4.5a or Q2 to Q5 in figure 4.5b), which will be referred to as `neighbour_node`:

2. For each `neighbour_node`:

   a) Loop over the edges that reach it according to the `edgnod` array, and count how many of its neighbours have been previously flagged.

   b) If the number of flagged nodes is greater than one, the element is not a quad. We can see it in figure 4.5a, where the flagged neighbouring nodes see each other. In that case, the diagonal node search is stopped. Otherwise, we must keep the diagonal nodes.

c) Loop again over the edges that reach `neighbour_node`, and store its neighbouring nodes (coloured in light green in figure 4.5c) in a `neighbour_list`.

3. Loop over the `neighbour_list` array and count how many times a node is stored. When a node appears twice, it is a diagonal node (the nodes D1 to D4, coloured in green in figure 4.5c, are the diagonal nodes of the Q1 node). The diagonal nodes are kept in an array analogous to `edgnod`, but for diagonal nodes instead of edges.

If the seed node has diagonal nodes attached, the basic agglomeration algorithm presented in the previous section is replaced by the following quad agglomeration algorithm:

1. Choose a seed point from the front of seed point candidates and agglomerate it, executing the operations specified in the second step of the basic agglomeration algorithm of section 4.1. The status of the agglomeration up to this point is represented in figure 4.6a, where a new coarse grid point, named Q1, and the edges Q1-C1, Q1-C2 and Q1-C3 and their associated areas, have been created.

2. Check if the seed point has diagonal nodes. In that case we must find a set of four available fine grid points to create an agglomerated quad:

   a) Loop over the diagonal nodes of the seed node. For each diagonal node that has not been previously agglomerated (node D1 in figure 4.6b) find which of its neighbours are coincident with the neighbours of the seed node (nodes N1 and N2 in figure 4.6b). We have a set of four nodes that constitute a coarse quadrilateral control volume resulting from the agglomeration of four quads from the fine grid. In the example of figure 4.6, these nodes are Q1, D1, N1 and N2.

   b) If a set of four nodes could not be found after looping over the seed node diagonal nodes, the quad agglomeration is not performed, and the basic agglomeration algorithm is executed. Otherwise, agglomerate the set of nodes, giving rise to a new coarse grid node Q1 (figure 4.6c). The front of seed node candidates and the edges of the coarse grid have also been updated.

3. If the quad agglomeration has been successful, the new seed point is not chosen according to the rules given in the previous section. Instead, a suitable seed point is sought to form a structured pattern of quads in the coarse grid. This seek is performed as follows:

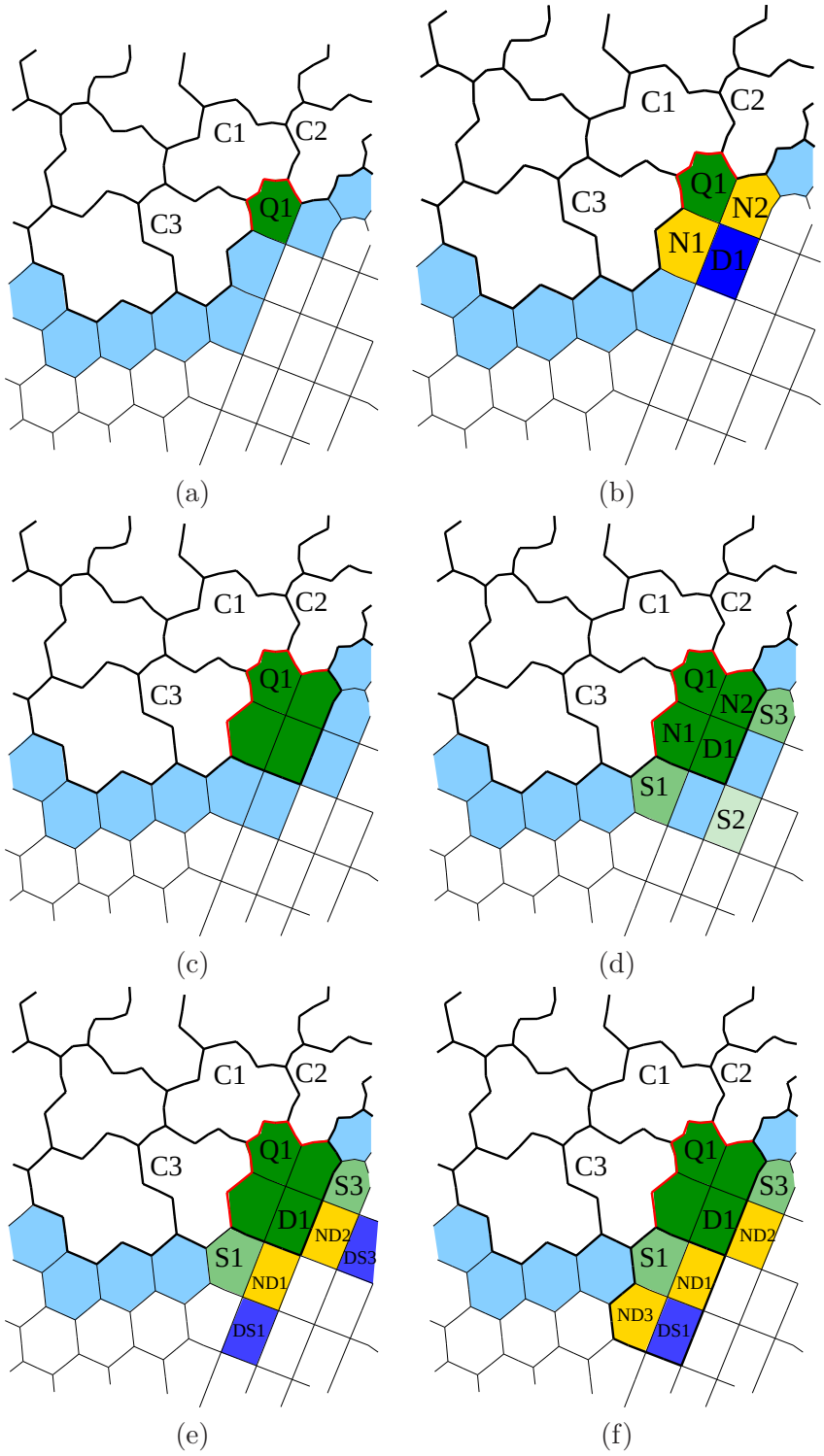


Figure 4.6: Description of the agglomeration process designed for quads. (a) A seed point that belongs to a quad is agglomerated. (b) Making use of the diagonal nodes, we find the nodes N1 and N2, which are neighbours of the Q1 and D1 nodes simultaneously, (c) An agglomerated quadrilateral control volume is formed. The new seed point is chosen to preserve the quadrilateral structure in the coarse mesh. (d) Keeping the nodes that are diagonal nodes of D1 and neighbours of N1 and N2 simultaneously (S1 and S3). (e) The new coarse grid quadrilateral control volume is formed with one of these nodes, with one of its diagonal nodes (the one that shares one neighbour with D1; ND1 and ND2), (f) and with their coincident neighbours, (ND1 and ND3). Steps (d) to (f) are repeated as long as the formation of coarse grid quads is possible.

a) Loop over the diagonal nodes of D1. For each diagonal node that has not been agglomerated yet (S1, S2 and S3 in figure 4.6d) find if any of its neighbouring nodes has been included in the previously agglomerated quad (N1 or N2, according to the figure). If this is the case, store the diagonal node (S1 and S3 in the example) in an array called `diagonal_list`.

b) Loop over the nodes in the `diagonal_list` array (S1 and S3). For each node:

   i. Find which of its neighbouring nodes is also neighbour of the D1 node (Nodes ND1 or ND2 in figure 4.6e). Store these nodes.

   ii. Loop over the S1 and S3 diagonal nodes and choose the one that shares one neighbour with the stored nodes ND1 and ND2 (DS1 and DS3 in figure 4.6e). At this moment we have kept three nodes for each possible coarse grid quadrilateral control volume: S1, DS1 and ND1 for one candidate, and S3, DS3 and ND2 for the other.

   iii. For the former candidate, the fourth node that is needed to obtain the new coarse grid control volume is the neighbour of DS1 and S1 simultaneously, which is not ND1. If this neighbouring node (ND3 in 4.6f) has not been previously agglomerated, we now have a set of four nodes to obtain the agglomerated control volume.

4. We repeat the previous point as long as there are fine grid quad nodes that can be agglomerated.

The idea is that once a quad has been agglomerated, the algorithm tries to form as many coarse neighbouring quads as possible. The use of this algorithm to agglomerate quads substantially improves the quality of the coarser dual mesh, as it is seen in figure 4.7. In it, we have plot a detail of a dual mesh that combines control volumes that come from triangles and quads (figure 4.7a). The result of agglomerating such mesh with the basic agglomeration step of section 4.1 and with the quad agglomeration algorithm may be seen in figures 4.7b and 4.7c. The quad coarsening algorithm shows greater grid regularity in the quad zone, yielding an increase in the fine to coarse grid edge ratio.

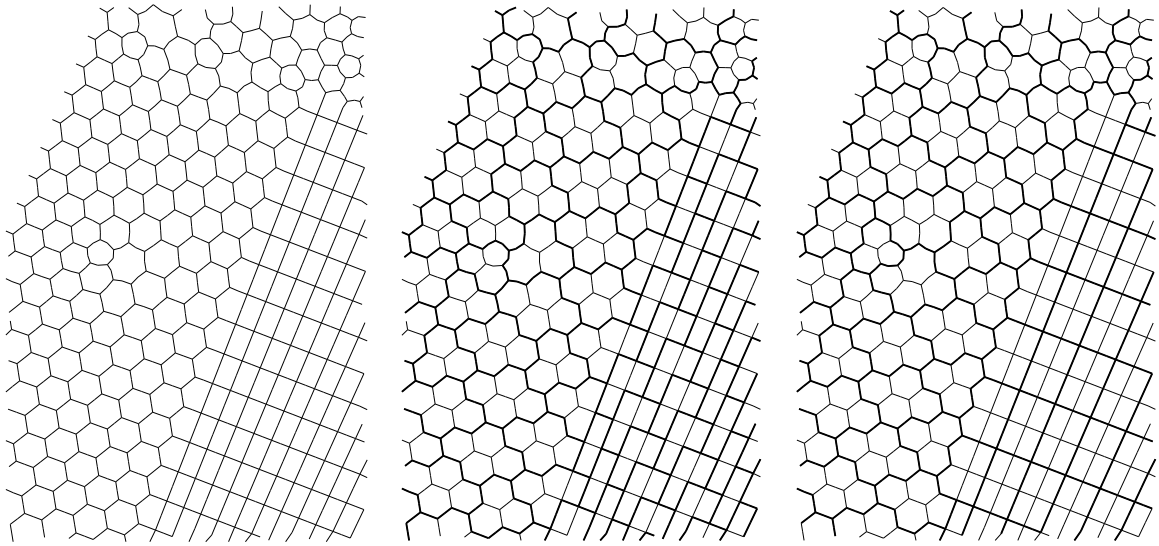


Figure 4.7: Comparison of the dual meshes obtained from an initial fine mesh (a) upon using the standard agglomeration algorithm (b) and the quad agglomeration algorithm (c).

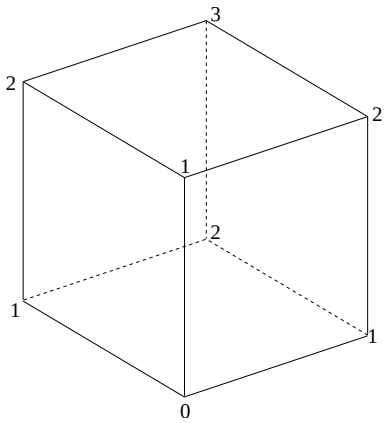


Figure 4.8: Sketch of the nomenclature employed in the hexahedron agglomeration algorithm.

4.3. Agglomerating hexahedra

The strategy to agglomerate quads outlined in section 4.2 fails when dealing with three-dimensional unstructured meshes made up of hexahedra. In three dimensions we aim at obtaining a structured pattern of hexahedra when agglomerating a grid made up of hexahedra. The problem is that the edge-based neighbours of node 0 (nodes 1 in figure 4.8) and its diagonal nodes (nodes 2) do not contain all the nodes in a hexahedron The array containing the diagonal nodes does not consider all the nodes in a hexahedron, as shown in figure 4.8. In this figure it is appreciated that the node 3 actually belongs to the hexahedron, but requires a special treatment to be taken into account when agglomerating such kind of elements. It is noted that the node 3 is the only one which is the neighbour of all the three diagonal nodes simultaneously. This fact can be used to build an additional logic to agglomerate hexahedra when a seed node is contained in such kind of element. The proposed algorithm does not conflict with the quad agglomeration, but it is an extension for three-dimensional cases. It consists of building groups of eight nodes that have not been previously agglomerated:

1. Loop over the diagonal nodes of the seed node (nodes 2). For each diagonal node, keep its neighbouring nodes in a list: `diagonal_neighbours`. This array has two entries: the index of the diagonal node and the indexes of its neighbouring nodes. Only the control volumes that have not been previously agglomerated must be in that list.

2. Loop over the entries of `diagonal_neighbours` and find which nodes are listed three times. We also obtain, as a by-product, the neighbours of such nodes, which are the diagonal nodes flagged with a '2' in figure 4.8.

3. The hexahedron is then formed with the following nodes:

   a) The seed node (node 0).

   b) Three neighbours of the seed node (nodes 1).

   c) Three diagonal nodes of the seed node (nodes 2).

   d) The node 3, which is the common neighbour of the three diagonal nodes.

Using this additional logic, the coarsening ratio in three-dimensional fully unstructured grids is, if not fully recovered (ideally an 8:1 agglomeration ratio is desired), at least increased.

4.4. Highly stretched meshes and directional agglomeration

When solving the Reynolds-Averaged Navier-Stokes equations, strong gradients of the variables appear in the normal wall direction, while along the wall the gradients are much smaller. These thin layers lead to the generation of highly stretched cells where the grid points are clustered in the normal wall direction. Wakes also exhibit the same features. These highly stretched cells cause problems when trying to damp high frequency errors by means of an iterative algorithm, since they are not as easily smoothed as in isotropic regions. This, in turn, degrades the convergence rate of multigrid methods. Directional agglomeration, often referred to also as semi-coarsening, is a mechanism to alleviate the problem, by agglomerating just in the normal direction. The effect is twofold: it reduces the aspect ratio of the cells in the coarser grids, and hence the stiffness associated with it and, since the mesh is not coarsened in the direction where the high frequency errors persist, these are left out of the multigrid strategy. These ideas are further explained in chapter 5, where the preconditioning of the RANS equations is addressed.

This agglomeration method requires the detection of the stretching direction. This is done with an algorithm that constructs lines that join the nodes in the normal direction [60]. Lines start from nodes with the most stretched cells and ends in zones where the cells are isotropic. The semi-coarsening is therefore applied just for the stretched cells, and not for the entire mesh, thus saving computing time and memory. The algorithm to construct these lines is as follows:

1. Loop over the edges of the grid and compute, for each node, the maximum, minimum and average values of the modulus of the areas associated to the edges that reach that node,

$$\begin{aligned} \sigma_i|_{max} &= max\left(|\sigma_{ij}|\right), \qquad j = 1, \#ed_i \\ \sigma_i|_{min} &= min\left(|\sigma_{ij}|\right), \qquad j = 1, \#ed_i \\ \overline{\sigma}_i &= \frac{1}{\#ed_i} \sum_{j=1}^{j=\#ed_i} |\sigma_{ij}| \end{aligned}$$

   and store the ratio of these values: $\sigma_i|_{max} / \overline{\sigma}_i$ and the two neighbouring nodes that yield the larger value of $\sigma_i|_{max}$ / $\sigma_i|_{min}$, that will be called `nearest_neighbours`.

2. Sort the nodes according to $\sigma_i|_{max} / \overline{\sigma}_i$, yielding a list where the nodes with larger $\sigma_i|_{max} / \overline{\sigma}_i$ are placed first.

3. Choose the first node of the sorted list, named `starting_node`. This node is the one that has the most stretched control volume, and hence it is an appropriate node to start the

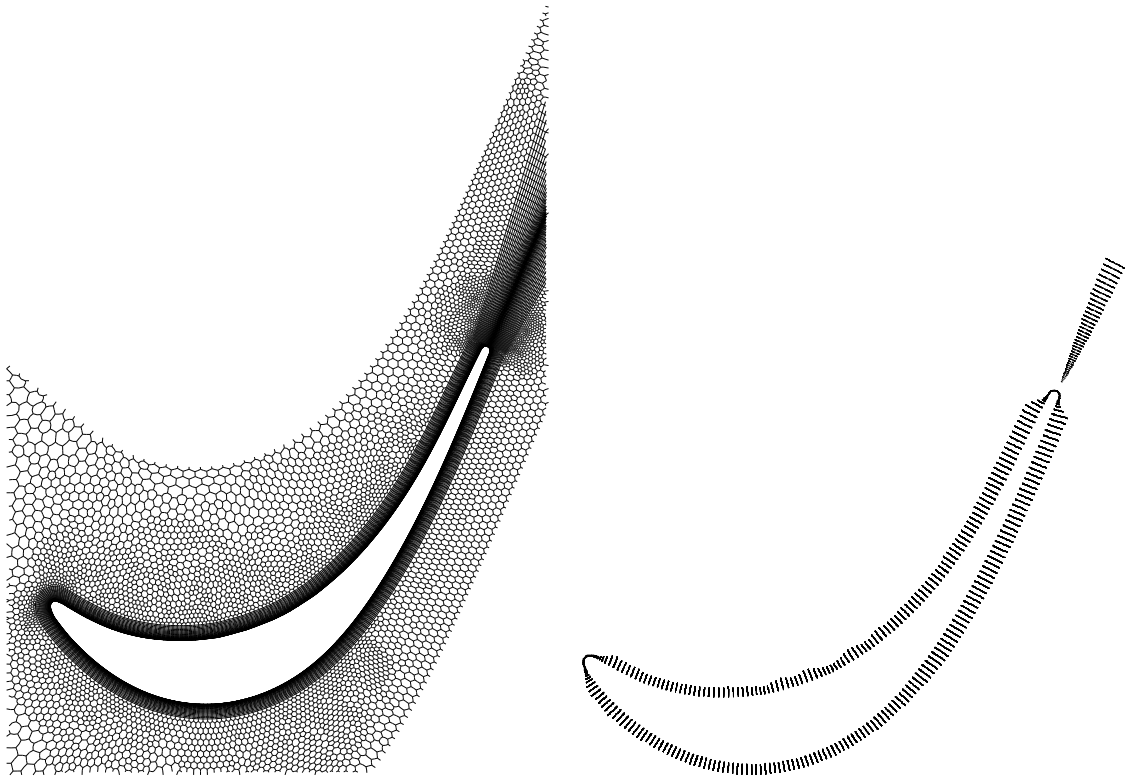


Figure 4.9: Left: Dual mesh of a T106 blade with stretched cells in the boundary layer and in the wake. Right: Line construction for semi-coarsening.

construction of the line.

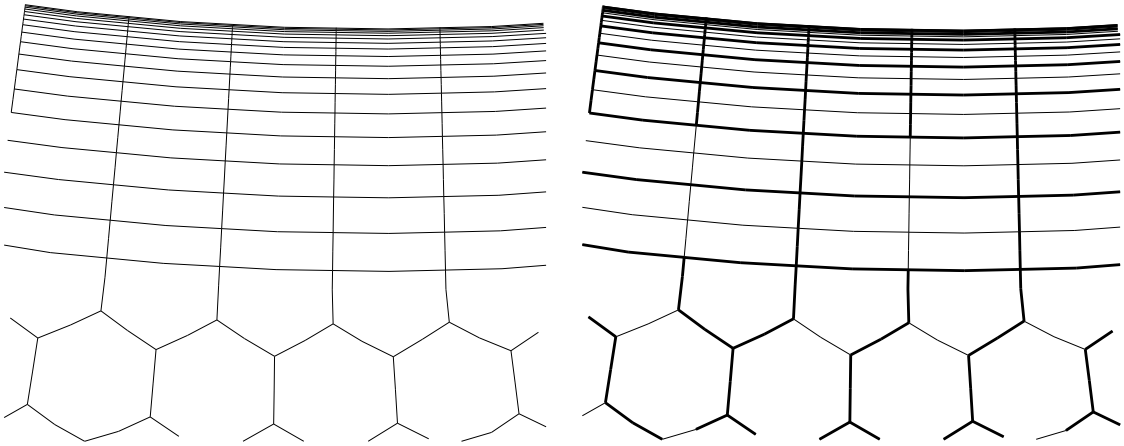
4. If $\sigma_i|_{max} \,/\, \sigma_i|_{min} \geq 4$, join the `starting_node` with one of the available `nearest_neighbours`. Set the `nearest_neighbour` node as the new `starting_node`.

5. Repeat step 4 while $\sigma_i|_{max} \,/\, \sigma_i|_{min} \geq 4$. The factor of four ensures that the control volumes that are being connected are stretched enough and that the line construction ends in a region with quasi-isotropic cells.

6. When no additional points can be added to the line, start building another line by choosing the first available node in the `sorted_list`, and repeat steps 4 and 5 until no more points can be added to a line. This way of choosing the first node of the line ensures that the line always begins in the highest stretched cell and ends in the isotropic zones.

The result of applying this algorithm is a set of lines like that depicted in figure 4.9 right. There, we present a detail of the dual mesh of a T106 airfoil with stretched cells in both the boundary layer and wake regions, and the lines that result upon the application of the previous algorithm.

The agglomeration is very simple once these lines have been constructed:



(a) (b)

Figure 4.10: Semi-coarsening example. (a) Detail of the boundary layer dual mesh of figure 4.9; (b) Semi-coarsened mesh.

1. Choose the first point of a line as the seed point and agglomerate it, creating a new coarse grid volume CL1.

2. Agglomerate just the `nearest_neighbour` of the seed point, if there is any. This neighbour is given by the relation constructed when executing the line construction algorithm presented above. If there is not any `nearest_neighbour`, then the semi-coarsening ends.

3. The new seed point is the `nearest_neighbour` of the node agglomerated in point 2. When we agglomerate it, we create a coarse volume CL2. We also construct a new line for the coarse grid, that joins the node CL1 to the CL2.

4. Repeat steps 2 and 3 for all the lines.

The result of applying this agglomeration algorithm is depicted in figure 4.10, where a detail of the dual mesh in the boundary layer region is plotted. It is seen in figure 4.10b that the mesh is just agglomerated in the stretching direction next to the wall, where the semi-coarsening agglomeration lines have been constructed. When the aspect ratio of the control volume areas drops below the value of 4 used to construct the lines, the line agglomeration algorithm is deactivated and we can see then that the quad agglomeration algorithm is used. Outside the quad mesh, the basic agglomeration algorithm is used.

4.5. Three-dimensional semi-unstructured agglomeration

Typical turbomachinery grids are constructed building a grid for a two-dimensional plane which is then extruded and deformed along the third dimension [17] (see figure 4.11a). The two-

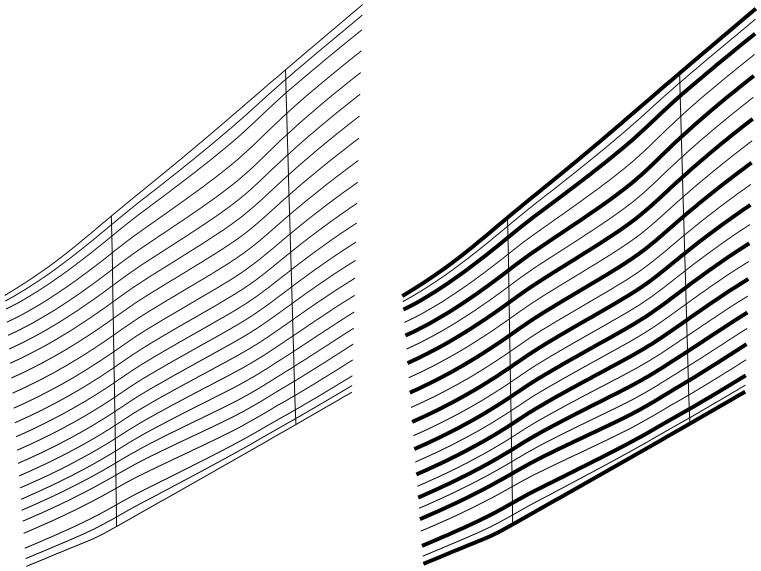


Figure 4.11: Meridional view of the dual mesh of a semi-unstructured grid (a) Fine grid, (b) Semi-coarsened mesh.

dimensional position of the nodes is adapted to the radial variation of the geometry but the grid connectivity is the same in each radial plane. This fact makes the three-dimensional agglomeration easier. We automatically find this extruded two-dimensional plane and store the correspondence between a node and its homologous in the upper plane. The algorithm to recognise the structured pattern needs that the pattern departs from one of the domain boundaries. Then the following operations are performed:

1. Loop over the boundaries. For each boundary:

   a) Initialise the number of levels of the structured pattern, `pattern_level` = 1.

   b) Loop over the nodes of the mesh. For each node whose level in the structured pattern is equal to `pattern_level`:

      i. Loop over its neighbouring nodes, making use of the `edgnod` array. For each neighbour, check if it belongs to the same level of the structured pattern, `pattern_level`, than the flagged node. Three cases may occur:

         - It belongs to the same level.
         - It belongs to a level immediately below the `pattern_level`, that is, to `pattern_level`− 1. In such case, we must check if the neighbour node is the homologous of the node we are working with.
         - It has not been classified yet. Then the neighbour is flagged as belonging to

`pattern_level` + 1.

c) If there is a structured pattern, all the neighbouring nodes belong to the same level except two, i.e.: the homologous of the working node in `pattern_level` − 1 and `pattern_level` + 1. If this condition is not fulfilled, stop the structured pattern search.

d) Count the nodes that belong to the `pattern_level` + 1.

e) If the number of such nodes is equal to the number of nodes in the boundary, increase the `pattern_level` counter by one and continue searching a new structured plane, repeating the steps (b) to (d). Otherwise, the search is stopped.

2. If the structured pattern search has been successful, for each mesh node we store two values: the `pattern_level` it belongs to and the index of the homologous node in the upper level. Thus, when agglomerating semi-unstructured meshes we perform the standard two-dimensional agglomeration outlined in sections 4.1, 4.2 and 4.4 for nodes that belong to `pattern_level` = 1. Once we have agglomerated the first level, we agglomerate the rest of them making use of the pattern information stored for each node. If the number of planes is n, then the number of layers is $n/2$ if n is even and $(n+1)/2$ if n is odd (figure 4.11b).

4.6. Agglomerating periodic nodes

Even though the seed points are chosen according to the priority list given in section 4.1, problems arise when dealing with typical turbomachinery flow domains, since in these domains periodicity relations must be preserved. Thus, when a periodic node is agglomerated, the same action must be performed with its periodic pair. Doing this action in the generic agglomeration causes problems with the coarse grid nodes numbering strategy and spoils the algorithm. We agglomerate periodic nodes first to avoid this problem. After all periodic nodes have been agglomerated, the standard agglomeration algorithm is recovered.

4.7. Complete agglomeration algorithm

The complete agglomeration algorithm is sketched in figure 4.12. The initial data of the unstructured fine grid that is agglomerated are the nodes and the edge connectivity, and the final result

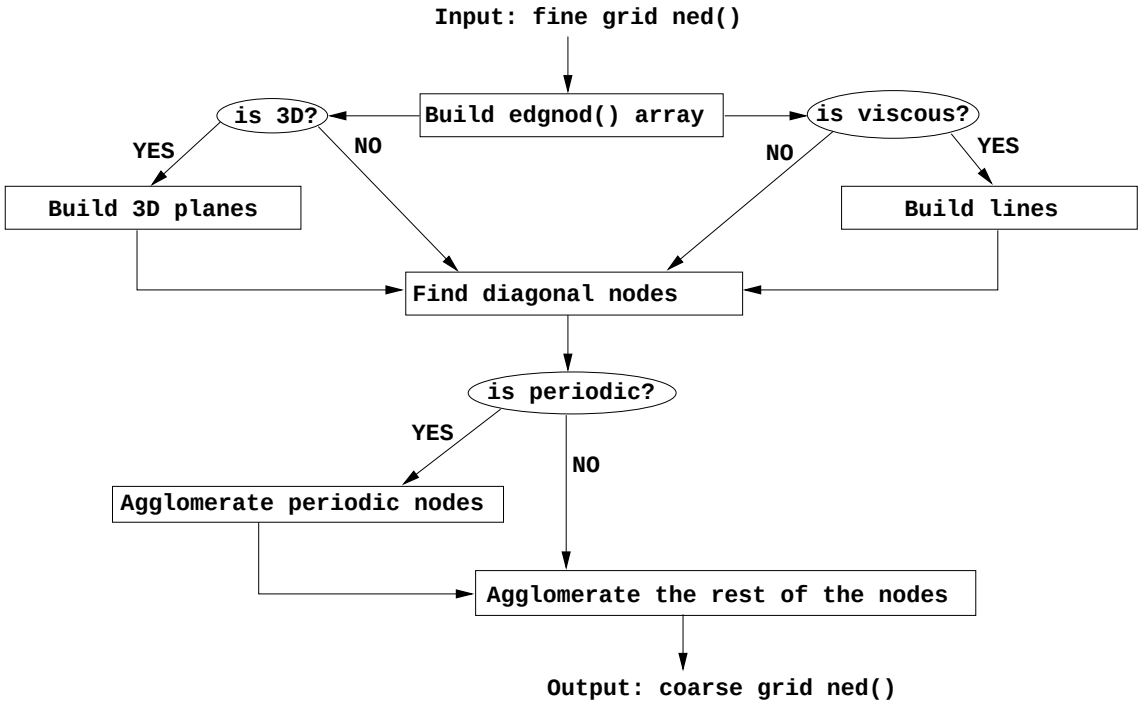


Figure 4.12: Sketch of the complete agglomeration algorithm. Departing from the edge connectivity of a fine grid, the agglomeration algorithm returns the edge connectivity of a coarser grid.

is an analogous set of data for the agglomerated mesh. The global sequence of steps followed to obtain a coarser mesh is the following:

1. As a preprocessing step, the `edgnod` array is calculated as specified in section 4.1.

2. If the case is three-dimensional, then try to recognise a two-dimensional pattern repeated along the direction of extrusion, as it has been explained in section 4.5.

3. If the case is viscous, there are stretched cells in the boundary layer and in the wake regions. To perform a semi-coarsening it these regions,we must construct the lines for the directional agglomeration outlined in section 4.4.

4. Find the diagonals of the mesh nodes, needed to correctly agglomerate quads performing the steps of the quad agglomeration algorithm of section 4.2.

5. If there exist periodic nodes, then agglomerate them first.

6. Finally, agglomerate the rest of the mesh nodes, by either performing the basic agglomeration, the quad agglomeration or the agglomeration for highly stretched cells.

After all volumes have been agglomerated, a new set of edge-data structure is returned, ready to be used with the same discretisation, and of course ready to perform additional agglomerations.

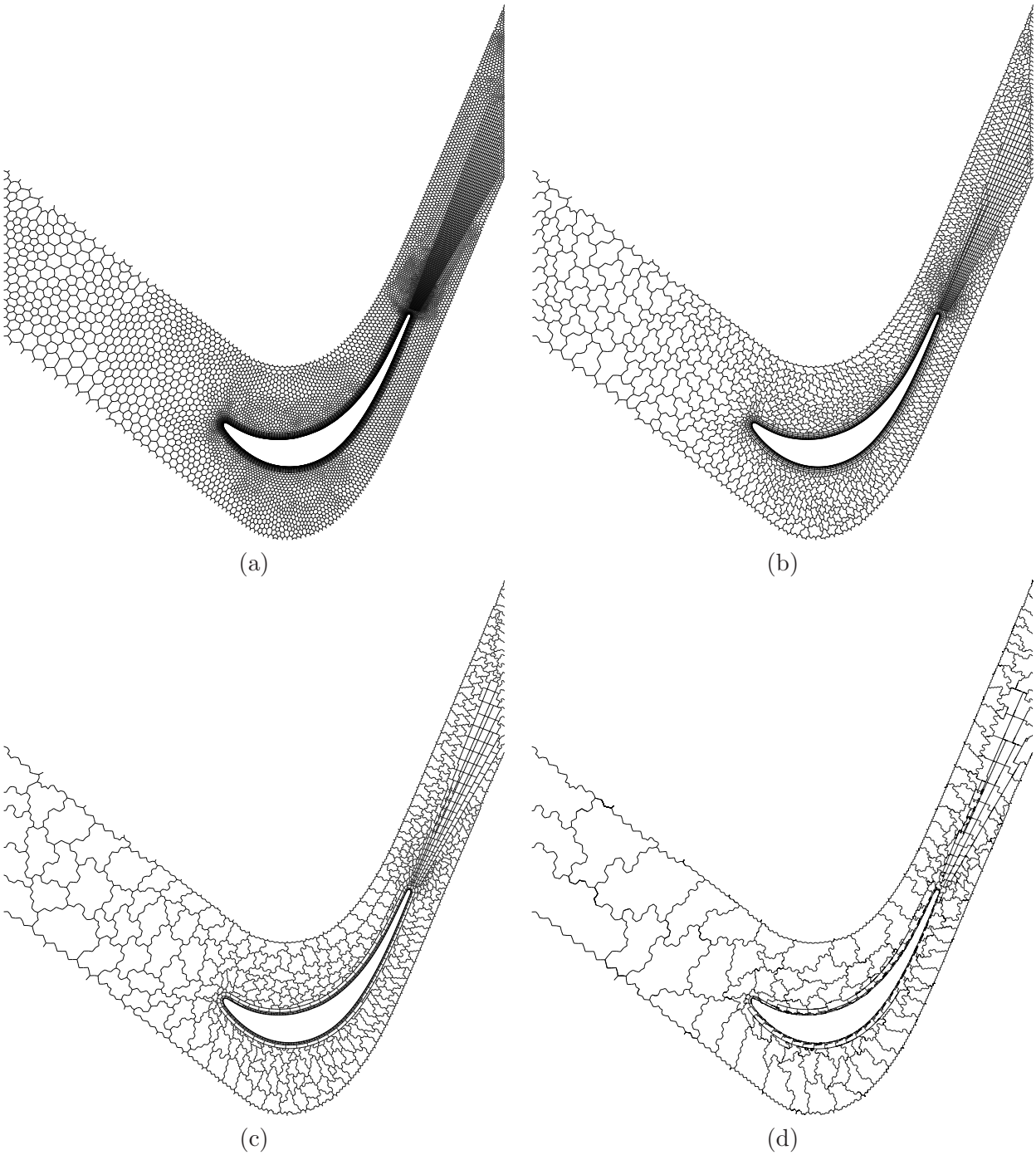


Figure 4.13: Grid agglomeration sequence on a T106 grid made of triangles. (a) Fine dual mesh. 11641 nodes; (b) First agglomerated level: 4019 nodes; (c) Second agglomerated level. 1479 nodes; (d) Third agglomerated level. 584 nodes.

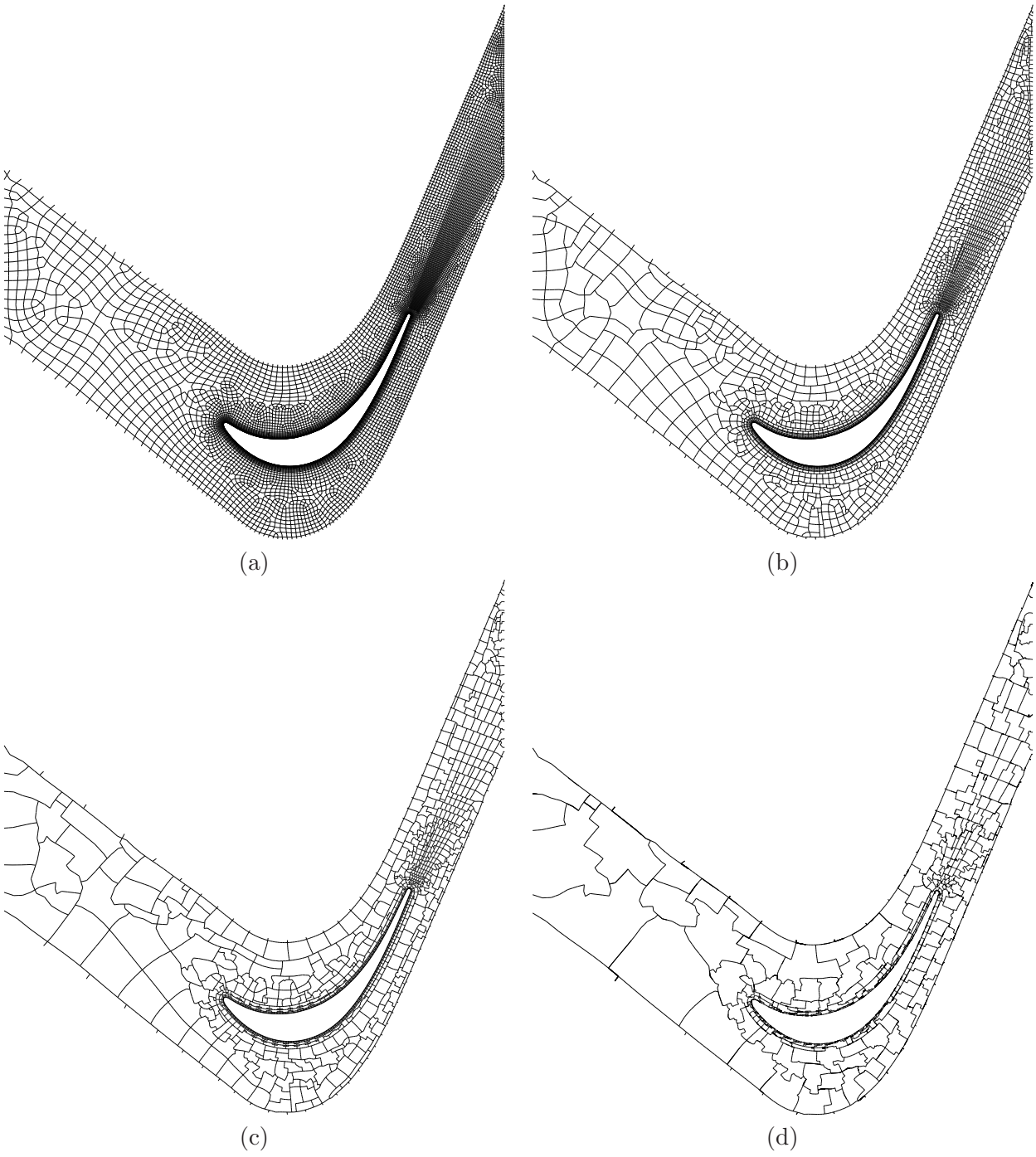


Figure 4.14: Grid agglomeration sequence on a T106 grid made of quads. (a) Fine dual mesh. 10964 nodes; (b) First agglomerated level: 3944 nodes; (c) Second agglomerated level. 1493 nodes; (d) Third agglomerated level. 588 nodes.

Two examples of agglomeration sequences can be seen in figures 4.13 and 4.14. These examples correspond to a two-dimensional domain of the T106 LPT airfoil[87], which have been meshed with triangles (figure 4.13) and quads (figure 4.14). The coarser meshes have been constructed using the agglomeration algorithms presented in the previous section. It is seen that the quad mesh structure of the fine mesh is almost conserved in the first agglomerated level for both cases, and is roughly conserved in the second agglomerated level. The third agglomerated level yields very irregular element shapes, but agglomerated meshes with less than 1000 points are rarely used, since its use does not improve the convergence rate. The effect of semi-coarsening in the highly stretched cells is also noticeable. The coarsening is performed just in the direction normal to the airfoil wall, and therefore the number of nodes in the airfoil boundary remains constant throughout the agglomeration process.

5. Preconditioning

Preconditioning techniques try to accelerate the convergence to the steady state in stiff problems. In the Navier-Stokes equations the stiffness is mainly due to the appearance of low Mach number regions. The problem could be solved by using an incompressible Navier-Stokes equations solver. However, the development of an specific tool for this type of flows is not usually desired, since typical turbomachinery applications have a much wider range of Mach numbers. Besides, low Mach number regions can also exist in compressible flows, as it is the case of the leading edge region and separation bubbles. The stiffness in these cases is provoked by the disparity of the system eigenvalues, since while acoustic waves are propagated with speed $u + c$ (where c is the speed of sound), entropy and vorticity waves are convected with the fluid speed $u \ll c$. This disparity causes the problem to be stiff. This issue is addressed changing problem eigenvalues, clustering them and thus making error smoothing techniques more effective. This is done by premultiplying the time derivative by a preconditioning matrix

$$P^{-1}\frac{d\mathbf{U}}{dt} + R\left(\mathbf{U}\right) = 0, \tag{5.1}$$

that tries to balance the propagation speed of all the waves. The preconditioning matrix modifies the physics of the problem, which is no longer time accurate. This effect makes the use of these techniques unfeasible for unsteady cases, but does not represent any problem if the aim is to reach the steady state, since when $d\mathbf{U}/dt = 0$, the effect of using the preconditioning matrix vanishes. Besides, convergence rate is greatly improved. A review of low Mach number preconditioning techniques can be found in [83].

The stiffness caused for the low Mach number regime is a physical stiffness, and is present in the resolution of the equations, whatever the discretisation is. However, when the continuous equations are discretised, other sources of stiffness may appear, the so-called discrete stiffness. This kind of stiffness arises when discretising boundary layers and other large gradient regions, which require a large amount of points in a privileged direction, but much less in every other.

This produces high aspect ratio cells that make the residual spatial operator eigenvalues cluster next to the phase-plane origin, where the temporal discretisation schemes have poor damping properties. This is solved again by premultiplying the time term with a P^{-1} matrix, as it is seen in Eq. (5.1). Now the aim of this premultiplication is not making the physical eigenvalues almost equal, but moving discrete eigenvalues away from the origin, where they will be appropriately damped by the smoother. This is accomplished by the so-called block-Jacobi preconditioning technique, developed either for structured [1, 72, 73, 75] or unstructured [63] meshes. With this method, eigenvalues are effectively moved away from very low damping regions.

This improvement of the smoother damping properties is essential when multigrid techniques are employed to solve the Navier-Stokes equations, since the multigrid algorithm requires very efficient smoothers capable of damping the oscillatory errors in each grid. Without this feature, the high frequency errors that are not damped in the fine grids turn smooth in coarser grids, where the smoother fails to damp them efficiently, and the multigrid algorithm looses all its effectiveness. The use of multigrid in conjunction with an appropriate preconditioning method has proven useful for solving the Navier-Stokes equations in a wide range of grids [73, 75, 72, 63, 60], without a significant degradation of the convergence rate due to the presence of high aspect-ratio cells.

This chapter analyses the improvements that result from the use of the preconditioning techniques in both low Mach number problems and highly stretched meshes.

5.1. Theoretical approach

The preconditioning matrices are based on the linearisation of the semi-discrete Navier-Stokes equations written in equation (2.7). The linearised inviscid and numerical diffusion fluxes are expressed as:

$$\mathbf{F}_{ij}^{I} = \frac{1}{2}\left[A_{ij}\left(\mathbf{u}_i + \mathbf{u}_j\right) - \left|A_{ij}\right|\left(-\frac{1}{2}\left(1-\kappa\right)\left(1-S\right)\left(L_i\left(\mathbf{u}\right) - L_j\left(\mathbf{u}\right)\right) + S\left(\mathbf{u}_i - \mathbf{u}_j\right)\right)\right] \tag{5.2}$$

A value of the switch $S = 1$ is used for the construction of the preconditioning matrices, which avoids including the contribution of the pseudo-Laplacian terms. It is a conservative choice that does not alter the effectiveness of the scalar nor the block-Jacobi preconditioning techniques very much[63].

The linearised viscous terms are expressed as

$$\mathbf{F}_{ij}^{V} = \frac{1}{2}\left[[LB]_{ij}\,(\mathbf{u}_i + \mathbf{u}_j) + B_{ij}\,\nabla\mathbf{u}|_{ij}\right] \tag{5.3}$$

where $\nabla\mathbf{u}|_{ij}$ is evaluated with the formula of equation (2.15).

Making use of the linearised matrices of Eqs. (5.2) and (5.3), the expressions of the scalar, block-Jacobi and low Mach number preconditioners are obtained in the following sub-sections.

5.1.1. Local time step

The scalar preconditioning is commonly known as the local time step. This acceleration technique consists in giving each cell in the discretised domain a different time step, in opposition to unsteady cases, where all the cells have a common time step, which is fixed by the smallest grid cell, since $\Delta t \propto \Delta x$. This scalar preconditioning is very efficient when dealing with cells with large size disparity, as it is the case when solving the Navier-Stokes equations, where typically the boundary layer cells are much smaller than those in the outer core flow, or the fluid domain boundaries are placed far away from the obstacles, as it occurs in external flows.

Thus, the local time step establishes a different time step for each cell according to its stability limit. This limit is given by the spectral radii of the jacobians of Eqs. (5.2) and (5.3):

$$\begin{aligned} \rho\,(A_{ij}) &= |\mathbf{v}_{ij}\cdot\mathbf{n}_{ij}| + c_{ij} \\ \rho\,(B_{ij}) &= \frac{\gamma}{\rho_{ij}}\left(\frac{\mu_{ij}}{Pr}\right)_{eq} \end{aligned} \tag{5.4}$$

Then the time step, for each cell, will be a combination of these values, which are separated into hyperbolic and parabolic parts.

$$\Delta t^{-1} = \Delta t_H^{-1} + \Delta t_P^{-1} \tag{5.5}$$

With the use of the local time step technique, the error waves are expelled out of the domain after a number of iterations proportional to the number of cells between the airfoil and the external boundaries.

The hyperbolic part of the time step is, for unstructured meshes with a median-dual grid with an edge-based data structure [37], by the expression

$$\Delta t_{H_i}^{-1} = \frac{1}{\vartheta_i}\frac{1}{CFL_H}\left[\sum_{j=1}^{\#ed_i}\left(|\mathbf{v}_{ij}\cdot\mathbf{n}_{ij}| + c_{ij}\right)\sigma_{ij} + \sum_{j=1}^{\#Bed_i}\left(|\mathbf{v}_{ij}\cdot\mathbf{n}_{ij}| + c_{ij}\right)\sigma_{ij}\right] \tag{5.6}$$

being σ_{ij} the area associated with the ij edge, σ_j the boundary area associated with the boundary edge j, and ϑ_i the volume associated to the node. Analogously, the parabolic part of the time step is

$$\Delta t_{P_i}^{-1} = \frac{1}{\vartheta_i} \frac{1}{CFL_P} \left[\sum_{j=1}^{\#ed_i} \frac{\gamma}{\rho_{ij}} \left(\frac{\mu_{ij}}{Pr} \right)_{eq} \frac{4}{|\mathbf{x}_j - \mathbf{x}_i|} \sigma_{ij} \right] \tag{5.7}$$

In the last expression, the gradient of the linearised conservative variables has been approximated by the expression:

$$\nabla \mathbf{u}_{ij} \simeq \frac{\mathbf{u}_j - \mathbf{u}_i}{|\mathbf{x}_j - \mathbf{x}_i|} \cdot \mathbf{l}_{ij}, \tag{5.8}$$

that is, only the second term of the right hand side of equation (2.15) is taken into account.

The factor of four in the equation (5.7) arises when considering the discretization of the second derivatives. A typical second order scheme gives:

$$\partial^2 u_j = u_{j+1} - 2u_j + u_{j-1}$$

If we try a solution

$$u_j = \widehat{u}_k e^{ikx_j}$$

and we introduce it on the second derivative we obtain, assuming Δx_j constant along the grid,

$$\partial^2 u_j = -2\widehat{u}_k e^{ikx_j} \left(1 - \cos k\Delta x\right)$$

which, for high frequency modes, $k\Delta x = \pi$, yields $\partial^2 u_j = -4\widehat{u}_k e^{ikx_j}$. In this expression we obtain the factor of four that we introduce in the parabolic part of the time step.

This scalar preconditioning is useful when dealing with non uniform meshes with large cell size disparities or with large spectral radius variations. Each cell runs with its own time step, which is chosen only on the basis of its geometric restrictions. Hyperbolic CFL limits the imaginary part of the eigenvalue, and the real part is limited by the parabolic one. However, the limitations posed by the use of the greatest eigenvalue of the jacobian matrix often lead to poor convection speed of the rest of the waves with smaller associated eigenvalues and, if used in conjunction with the scalar artificial dissipation, to an excess of dissipation in such waves that affects the desired precision of the final solution.

5.1.2. Block-Jacobi Preconditioning

Although the scalar preconditioning reduces one of the main sources of stiffness, namely the cell size variation along the domain, it is not able to deal with mesh stiffness associated to the cell stretching, that may be encountered in the boundary layer region, where points are typically clustered in the wall normal direction. As pointed out in the introduction, increasing the cell aspect ratio produces a clustering of some eigenvalues next to the origin of the Fourier space. This effect makes impossible for the smoother to damp certain high frequency modes appropriately and therefore the convergence rate is significantly degraded.

This effect is particularly important when the multigrid technique is used to improve the convergence rate. Multigrid benefits are based on the ability of the smoother to damp high frequency errors. If this ability is disabled somehow, as in the presence of stretching or low Mach number flows, the whole algorithm fails to converge, since high frequency errors are aliased to low frequency ones when moving to coarser grids, and the smoother does not damp them at all. Keeping this idea in mind, it is clear that the preconditioning technique must be a suitable technique for its use in conjunction with a multigrid strategy. Allmaras [1] demonstrated that the block-Jacobi preconditioning technique is efficient in terms of clustering away the eigenvalues from the origin of the Fourier space. Pierce *et al.* [75] demonstrated that for very high aspect ratio cells ($\lambda \rightarrow \infty$), not all high frequency modes are properly damped, particularly those modes which are high frequency in streamwise direction and low frequency in the other. This causes the convergence rate to degrade because multigrid can not cope with that high frequency appropriately. This behaviour can be avoided if a semi-coarsening multigrid strategy is implemented in the solver. The grid is not coarsened in the streamwise direction, therefore aliasing in that direction is avoided and the whole scheme recovers one of its fundamental goals, which is not to pass high frequency errors to coarser grids. Since the full semi-coarsening is very expensive in terms of both computation time and memory requirements, directional agglomeration is only used in high aspect ratio cells [75, 60]. The combination of preconditioning and directional agglomeration has proven successful in improving convergence rates, and is at the same time easy to parallelise.

Extracting the central node terms in both inviscid (Eq. (5.2)) and viscous (Eq. (5.3)) flux

contributions leads to the block-Jacobi preconditioning formula for unstructured grids:

$$P_{Ji}^{-1} = \frac{1}{\vartheta_i}\frac{1}{CFL_H}\left[\sum_{j=1}^{\#ed_i}\left(\frac{1}{2}\left|A_{ij}\right| + B_{ij}\frac{1}{|\mathbf{x}_j - \mathbf{x}_i|}\right)\sigma_{ij} + \sum_{j=1}^{\#Bed_i}\frac{1}{2}\left|A_{ij}\right|\sigma_{ij}\right] \tag{5.9}$$

where the same gradient approximation of equation (5.8) holds for this case. It is seen that a matricial numerical dissipation term is needed to correctly implement the block-Jacobi preconditioning. If scalar dissipation is used, the block-Jacobi preconditioning matrix is reduced to the local time step, since $|A_{ij}|$ reduces to $(|\mathbf{v}_{ij}\cdot\mathbf{n}_{ij}| + c_{ij})\cdot I$. The matrices $|A_{ij}|$ and B_{ij} are obtained in appendix A.

If we neglect the contribution of B_{ij} to the preconditioning matrix, which is usually small in zones where the viscous terms are not dominant, the matrix P_{Ji}^{-1} is singular at both $M = 0$ and $M = 1$ conditions, where the eigenvalues λ_{ij} associated to either the convective or one of the acoustic waves vanish. This makes the system of equations unstable, since very large increments of P_{Ji}^{-1} are obtained in these two limits. This problem is solved by limiting the minimum value of the eigenvalue, incorporating an entropy fix. This fix is different from that introduced in the artificial dissipation terms [53], since the minimum value allowed is greater due to stability restrictions [39]. The expression for this fix is:

$$\left|\overline{\lambda}_{ij}\right|^* = \begin{cases} \left|\overline{\lambda}_{ij}\right|, & \left|\overline{\lambda}_{ij}\right| \geq \varepsilon_{ij} \\ \dfrac{\left|\overline{\lambda}_{ij}\right|^2 + \varepsilon_{ij}^2}{2\varepsilon_{ij}}, & \left|\overline{\lambda}_{ij}\right| < \varepsilon_{ij} \end{cases} \tag{5.10}$$

being $\varepsilon_{ij} = \dfrac{c_{ij}}{4}$ and $\left|\overline{\lambda}_{ij}\right| = max\left(\dfrac{|\lambda_i + \lambda_j|}{2}, K\left|\lambda_j - \lambda_i\right|\right)$, where K is a constant chosen to be 2.

When the $k-\omega$ turbulence model is used, we must define a strategy to include the new variables in the iterative method. If the local time step is chosen as a preconditioner, the $k-\omega$ equations are advanced in time with the same time step than the rest of equations. However, if we use the block-Jacobi preconditioning technique, we should couple the $k-\omega$ equations with the Reynolds-Averaged equations and rewrite the block-Jacobi preconditioning formulation including the additional terms resulting from the coupled linearisation. This would require more memory, since a 7×7 matrix should be stored for each node instead of the standard 5×5 matrix, and also a CPU cost increase, since inverting a larger matrix requires more operations. To overcome these drawbacks, these equations are treated separately. Based on the fact that the k and ω variables are convected in first approximation with speed u, we compute a local

	$m\Delta x$	$n\Delta y$
H_xH_y	$[\pi/2, \pi]$	$[\pi/2, \pi]$
L_xH_y	$[0, \pi/2]$	$[\pi/2, \pi]$
H_xL_y	$[\pi/2, \pi]$	$[0, \pi/2]$

Table 5.1: Wave numbers for each high frequency rank.

time step for these equations, analogous to that of Eq. (5.6), but with an eigenvalue $|\mathbf{v}_{ij} \cdot \mathbf{n}_{ij}|$ instead of $|\mathbf{v}_{ij} \cdot \mathbf{n}_{ij}| + c_{ij}$.

5.2. Stability analysis

The analysis of the previously described preconditioning techniques consists in the representation of the Fourier footprints of the eigenvalues corresponding to the matrix that results from the application of either the local time step or the block-Jacobi preconditioning to the spatial discretisation. The linearised fluxes (Eqs. (5.2) and (5.3)) are written for a two-dimensional equispaced structured mesh and perturbed about a uniform steady solution with an harmonic flow field that is represented, in a simplified notation, by $\mathbf{u}_{ij} = \widehat{\mathbf{u}}_{mn}(t)\, e^{i(mx_i + ny_j)}$. We obtain the following discrete spatial operator

$$
\begin{aligned}
Z \;=\;& \left[|A|_x \, i \sin m\Delta x - S\,|A|_x \,(1-\cos m\Delta x) + \right. \\
+\;& \left. (1-S)\,\frac{1-\kappa}{2}\,|A|_x \,(2-\cos m\Delta x - \cos n\Delta y)\,(1-\cos m\Delta x) \right] \cdot \Delta y + \\
+\;& \left[|A|_y \, i \sin n\Delta y - S\,|A|_y \,(1-\cos n\Delta y) + \right. \\
+\;& \left. (1-S)\,\frac{1-\kappa}{2}\,|A|_y \,(2-\cos m\Delta x - \cos n\Delta y)\,(1-\cos n\Delta y) \right] \cdot \Delta x \\
-\;& B_x|_x \,(1-\cos m\Delta x)\,\frac{\Delta y}{\Delta x} - B_y|_y \,(1-\cos n\Delta y)\,\frac{\Delta x}{\Delta y} - \\
-\;& \left(B_x|_y + B_y|_x \right) \sin m\Delta x \sin n\Delta y
\end{aligned}
\tag{5.11}
$$

The eigenvalues of the $P \cdot Z/\vartheta_i$ operator, being P either the local time step or the block-Jacobi preconditioning method, provide us the damping of the error modes that are present in our discretised domain. We have special interest in those modes which are high frequency in at least one of the grid directions, because these modes need to be well damped if a multigrid technique is to be used efficiently. We are representing these modes as H, or high frequency modes, and L, or low. Thus, the H_xH_y modes will account for those which are high frequency in both x and y directions. The other two families of interesting modes are, then, the H_xL_y and the L_xH_y, which are high frequency modes in at least one of the two directions. The wave numbers represented

in all the figures of this chapter are specified in table 5.1.

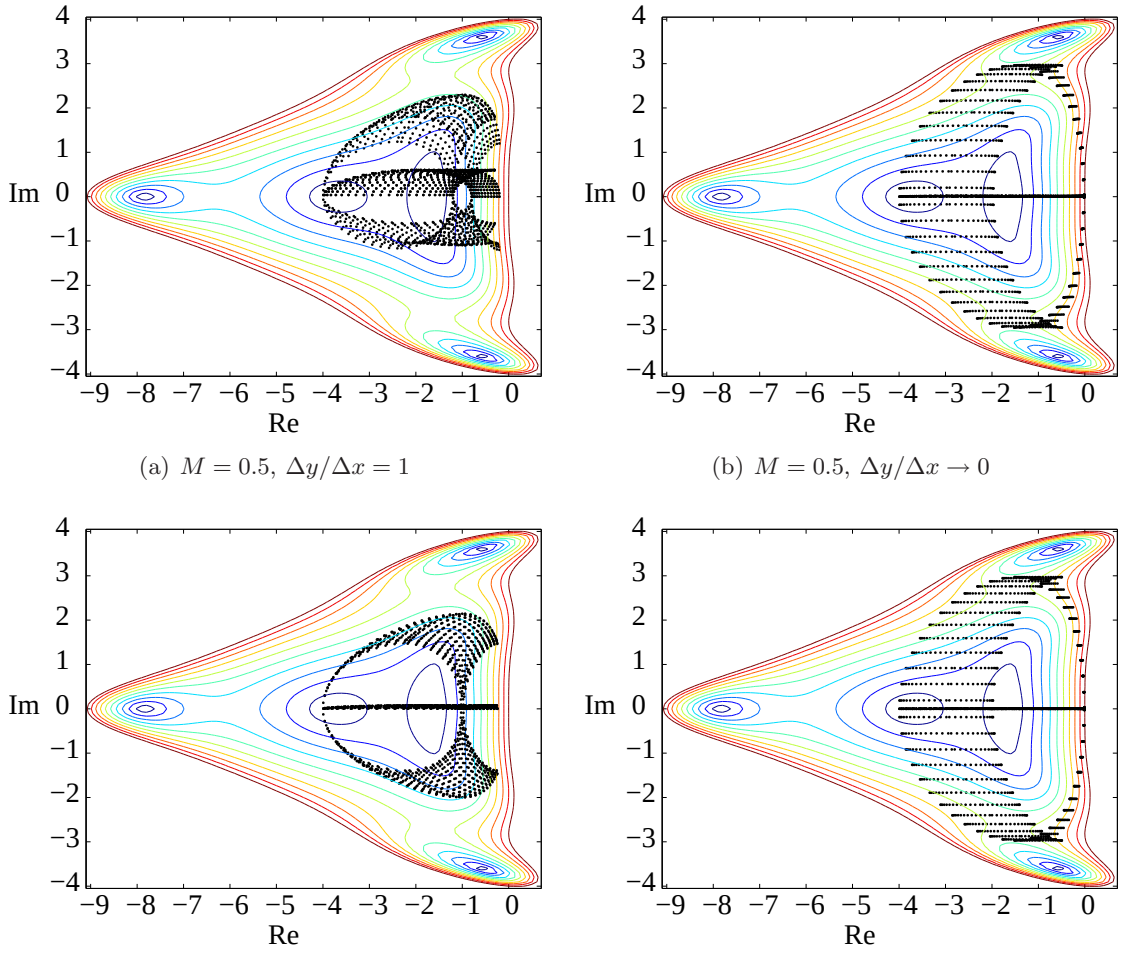
The design of the numerical scheme is multigrid oriented, therefore a good smoother is sought. The preconditioning method must show the ability to cluster all the eigenvalues away from the origin of the Fourier space, where the damping per time step is very high. Three cases are presented:

1. Scalar artificial dissipation with local time step.
2. Matricial artificial dissipation with local time step.
3. Matricial artificial dissipation with block-Jacobi preconditioning.

The parameters of study are the Mach, Reynolds and Prandtl numbers and the grid parameters, such as the grid aspect ratio $\Delta y/\Delta x$, and the angle between the velocity and the cell orientation, $\alpha = \arctan(v/u)$. We are studying four configurations:

1. $M = 0.5$ and isotropic cells with $\Delta y/\Delta x = 1$
2. $M = 0.5$ and stretched cells with $\Delta y/\Delta x \to 0$. Actually a value $\Delta y/\Delta x = 0.01$ has been used.
3. $M = 0.05$ and isotropic cells with $\Delta y/\Delta x = 1$
4. $M = 0.05$ and stretched cells with $\Delta y/\Delta x = 0.01$

The low Mach number cases have been first analysed without making use of the low Mach number preconditioning technique, that is addressed separately. A value of $\alpha = 0$ has been used, since it is the most restrictive in terms of eigenvalue clustering to the origin, even though the dependence on this parameter has also been checked for the most interesting cases. The grid Reynolds number, $Re_\Delta = (u\Delta x + v\Delta y)/\nu$, has been set large enough to override its influence, that fades the effects of the preconditioning techniques. All cases have been plotted for $CFL = 6$, $\kappa = 2/3$, using just the fourth differences in the numerical diffusion terms (a value $S = 0$ of the artificial viscosity switch is used in Eqs. (5.2) and (5.11)). The Fourier footprints of the high frequency spatial eigenvalues of table 5.1 are plotted together with the iso-damping contours of a 5-stage Runge-Kutta scheme described in section 2.5 for each of the three cases enumerated above.



(a) $M = 0.5$, $\Delta y/\Delta x = 1$

(b) $M = 0.5$, $\Delta y/\Delta x \to 0$

(c) $M = 0.05$, $\Delta y/\Delta x = 1$

(d) $M = 0.05$, $\Delta y/\Delta x \to 0$

Figure 5.1: Influence of the Mach number and grid stretching on the eigenvalue footprint of the spatial discretisation. Scalar artificial dissipation with local time step.

5.2.1. Scalar artificial dissipation and local time step

The scalar artificial dissipation formulation is directly derived from the matricial artificial dissipation formulation when substituting $|A|$ in Eq. (5.11) by an identity matrix multiplied by the spectral radius of $|A|$, namely $|\mathbf{v} \cdot \mathbf{n}| + c$. Since the block-Jacobi preconditioning matrix of equation (5.9) is obtained by extracting the terms that affect the central node of the discretisation, in the case of the scalar artificial dissipation this matrix is also reduced to $(|\mathbf{v} \cdot \mathbf{n}| + c)\, I$ (neglecting the contribution of the viscous terms), hence transforming this preconditioning technique into the well known local time step technique.

Figure 5.1 shows the Fourier footprint of the high frequency spatial eigenvalues obtained using the scalar artificial dissipation and local time step. The case with isotropic grids and $M = 0.5$ (fig. 5.1a), shows a good clustering of the eigenvalues away from the origin of the Fourier space. The discretization does not introduce any kind of stiffness, and the only source of stiffness would

be the case with low Mach number (figure 5.1c).

When $M = 0.05$, there are two families of high frequency error modes, namely the entropy and vorticity modes, that are loosely propagated, i.e., their imaginary part is close to zero, even though their real part has much larger values. The reason of this behaviour is more clearly seen if the expressions of the local time step (Eq. (5.6)) and the discrete spatial operator (Eq. (5.11)) are written in the limit of very low Mach number, i.e., $u, v \ll c$:

- The local time step yields

$$\Delta t = \frac{CFL \cdot \vartheta_i}{2c\,(\Delta y + \Delta x)} \tag{5.12}$$

  where the contribution of the viscous terms has been neglected since $Re_\Delta \to 0$. The two-dimensional control volume $\vartheta_i = \Delta x \Delta y$.

- The discrete spatial operator is

$$\begin{aligned} Z &= T_x \left[\Lambda_x \cdot i \sin m\Delta x - |\Lambda_x| \left(1 - \cos m\Delta x\right)\right] T_x^{-1} \cdot \Delta y + \\ &+ T_y \left[\Lambda_y \cdot i \sin n\Delta y - |\Lambda_y| \left(1 - \cos n\Delta y\right)\right] T_y^{-1} \cdot \Delta x \end{aligned} \tag{5.13}$$

  where we have written only the second differences term of the numerical diffusion for the sake of clarity, and we have also neglected the contribution of the viscous terms. T is the matrix resulting from the diagonalisation of A.

The matrices Λ_x and Λ_y contain eigenvalues much smaller that the speed of sound, i.e., those corresponding to the entropy and vorticity waves. For the scalar numerical diffusion formulation, the eigenvalues of the $|\Lambda_x|$ and $|\Lambda_y|$ matrices are equal to the largest of them ($|\Lambda| = c \cdot I$) . The eigenvalues of the product $\Delta t \cdot Z/\vartheta_i$ have their real part correctly scaled, of order unity, since the local time step is proportional to $1/c$ and the real part of the eigenvalues is proportional to c. However, the imaginary part of the entropy and vorticity eigenvalues is almost zero, yielding a very small phase velocity.

Thus, the entropy and vorticity waves are excessively dissipated. Although we are only presenting results for the high frequency errors, the situation is analogous for the low frequency waves, which we are trying to correctly represent in our simulations. This situation is fixed using a low Mach number preconditioning technique, which will be addressed in section 5.3.

The use of high aspect ratio cells (figures 5.1b and 5.1d) generates two main modifications in the footprint, whatever the Mach number is. First, some of the eigenvalue families are clustered

next to the origin, and second many of them are very close to the real axis, meaning that the flow variables that represent are convected with a velocity close to zero. The reason for that behaviour is deduced with an analysis similar to that performed for the low Mach number case, but now with $\Delta y/\Delta x \ll 1$. The assumptions made for the previous analysis, namely that $Re_\Delta = 0$ and that only the second differences of the numerical diffusion formulation are retained, hold for this case.

- The local time step for highly stretched cells is

$$\Delta t = \frac{CFL \cdot \vartheta_i}{2c\Delta x} + \mathcal{O}\left(\frac{\Delta y}{\Delta x}\right) = \frac{CFL \cdot \Delta y}{2c} + \mathcal{O}\left(\frac{\Delta y}{\Delta x}\right)$$

- The discrete spatial operator is

$$Z = T_y \left[\Lambda_y \cdot i\Delta x \cdot \sin n\Delta y - |\Lambda_y| \left(1 - \cos n\Delta y\right) \cdot \Delta x + \mathcal{O}\left(\frac{\Delta y}{\Delta x}\right)\right] \cdot T_y^{-1}$$

Therefore, the eigenvalues of $\Delta t \cdot Z/\vartheta_i$ are, in first approximation,

$$\frac{CFL}{2} \left\{ \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \end{bmatrix} \cdot i \sin n\Delta y - \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} (1 - \cos n\Delta y) \right\} \tag{5.14}$$

since we have assumed that $\alpha = 0$ and hence $v = 0$. The values of the previous expression represent, from top to bottom, the eigenvalues of the entropy, vorticity, and acoustic waves.

It is observed in Eq. (5.14) that the eigenvalues do not depend on the velocity u. That explains why the eigenvalue footprint for stretched cells does not depend on the Mach number (figures 5.1b and 5.1d are identical). It is also observed that many eigenvalues of the $H_x L_y$ waves lie next to the origin of the Fourier space, where the damping of the Runge-Kutta iterative scheme is very small. This effect makes the multigrid efficiency decay, and should be repaired for optimal convergence rate. Besides, the use of scalar artificial dissipation introduces an excess of dissipation in the entropy and vorticity modes, hence the number of points placed in the boundary layer normal direction should be very large to correctly resolve the low frequency modes of the entropy and vorticity. Otherwise, the damping associated to these waves, that are not propagated, introduces large errors in the resolution of the flow field in the boundary

layer. The use of a matricial artificial dissipation model, such as the Roe's formulation of the numerical fluxes [78, 53], solves this problem.

5.2.2. Matricial artificial dissipation and local time step

In this formulation the matrix of the numerical diffusion, $|A|$, is given by equation (A.1), and the dissipation is fixed independently for each characteristic (entropy, vorticity and acoustics). The use of the matricial dissipation avoids the overdamping of the entropy and vorticity waves in stretched cells, and the overdamping of the entropy wave in $M \ll 1$.

For $M \ll 1$ the eigenvalues are similar to those of the scalar artificial dissipation formulation, but their real part is modified due to the use of a matricial numerical diffusion. The eigenvalues of $|A|$ are now $|\Lambda| = [|u| \;\; |u| \;\; |u+c| \;\; |u-c|]$. The eigenvalues resulting from the product of the discrete spatial operator of Eq. (5.13) and the time step of Eq. (5.12), $\Delta t \cdot Z/\vartheta_i$, are also modified due to the use of the matricial artificial dissipation. The eigenvalues of the acoustic modes are properly scaled, but those of the entropy and vorticity are not.

The eigenvalues of entropy modes are collapsed into the origin of the Fourier space, i.e., these modes are not damped nor propagated. That is avoided with the entropy fix, that limits the minimum eigenvalue, yielding a modified $|\Lambda^*| = [|u|^* \;\; |u|^* \;\; |u+c| \;\; |u-c|]$, where $|u|^*$ is provided by Eq. (5.10).

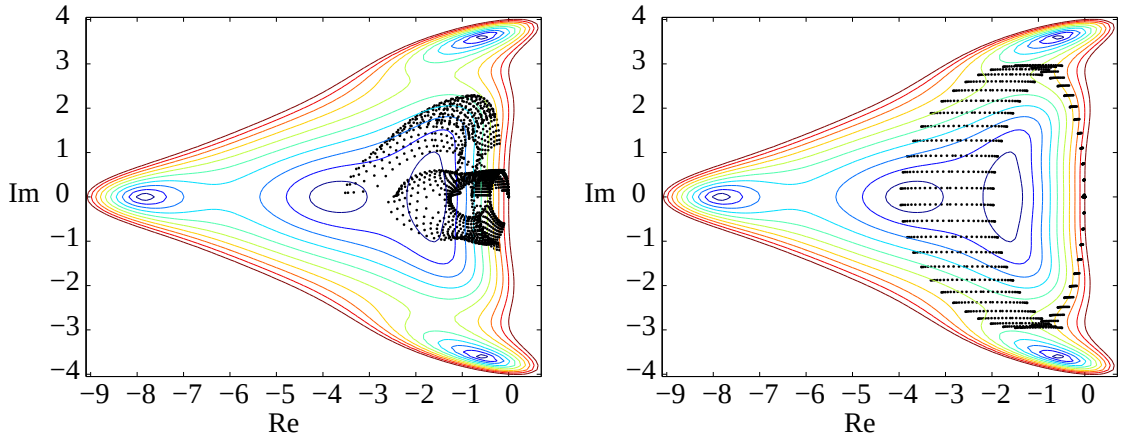
The eigenvalues of the vorticity have appropriate damping factors, but they are not propagated. This behaviour produces an excess of dissipation for this family of modes, analogous to that explained in section 5.2.1 for the entropy and vorticity modes when $M \ll 1$.

For $\Delta y/\Delta x \ll 1$ the real part of the eigenvalues written in Eq. (5.14) is also modified:

$$\frac{CFL}{2}\left\{\begin{bmatrix}0\\0\\1\\-1\end{bmatrix}\cdot i\sin n\Delta y - \begin{bmatrix}M^*\\M^*\\1\\1\end{bmatrix}(1-\cos n\Delta y)\right\} \tag{5.15}$$

where $M^* = |u|^*/c$. All modes of the entropy and vorticity waves are also collapsed into the origin of the Fourier space if the entropy fix is not used (if $M^* = 0$), whatever the Mach number, and most of the $H_x L_y$ modes of the acoustic waves lie next to the imaginary axis, where the damping is low.

All these effects can be appreciated in figure 5.2, that represents the eigenvalue footprint for



(a) $M = 0.5,\ \Delta y/\Delta x = 1$

(b) $M = 0.5,\ \Delta y/\Delta x \to 0$

(c) $M = 0.05,\ \Delta y/\Delta x = 1$

(d) $M = 0.05,\ \Delta y/\Delta x \to 0$

Figure 5.2: Influence of the Mach number and grid stretching on the eigenvalue footprint of the spatial discretisation. Matricial artificial dissipation with local time step.

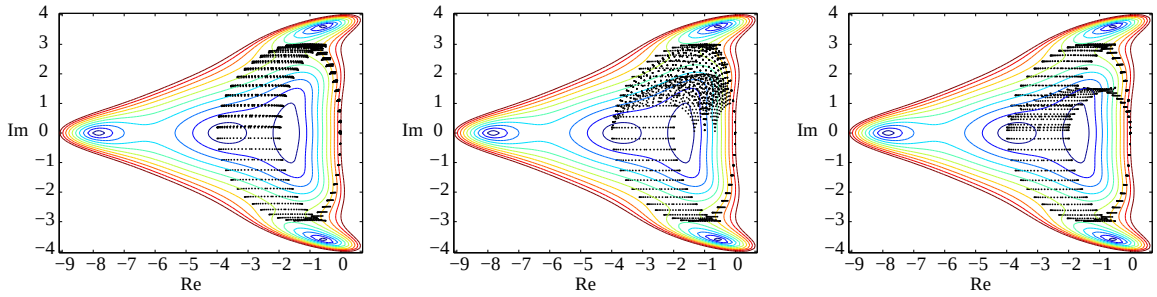
this case. We have also represented them for two Mach numbers and for isotropic and highly stretched cells. In the case with isotropic cells and $M = 0.5$ (figure 5.2a), some of the footprints lie very close to the origin, thus providing poor damping properties. That is produced because there are smooth modes in x direction ($k\Delta x = 0$) where the discrete spatial operator contains just terms coming from the A_y and $|A_y|$ matrices. The entropy and vorticity eigenvalues of these matrices vanish, since $v = 0$ when the flow is aligned with the cell. This effect is also corrected with the use of the entropy fix. Despite this special case, most of the eigenvalues lie far away from the origin, hence the damping for these modes is appropriate.

When $M = 0.05$ the vorticity eigenvalues are attached to the real axis. They have appropriate damping values but are hardly propagated due to the low Mach number. The entropy eigenvalues lie very close to the Fourier space origin, hence they are not propagated nor damped.

The situation is aggravated when high aspect ratio cells are considered (figures 5.2b and 5.2d). The situation presented when discussing the scalar artificial dissipation case (figures 5.1b and 5.1d) is partially corrected with the use of the matricial artificial dissipation, avoiding the excess of dissipation in stretched cells, but now the entropy and vorticity eigenvalues have very low values of both their real and imaginary parts. The real part of the $H_x L_y$ acoustic modes is also very low, according to Eq. (5.15) and hence they are just slightly damped. Most of the eigenvalues lie inside the minimum damping zone, close to the Fourier space origin, and the smoother is unable to efficiently remove these error modes. Therefore high frequency oscillatory errors cannot be damped and the efficiency of the multigrid algorithm decreases unless an appropriate technique, such as the block-Jacobi preconditioning, is used.

5.2.3. Matricial artificial dissipation and block-Jacobi preconditioning

As pointed out in section 5.2.1, the use of the matricial formulation of the artificial dissipation yields the expression of the block-Jacobi preconditioning matrix of Eq. (5.9), since this matrix is obtained by extracting the terms affecting the central node of the discretisation of Eq. (5.2). Even though the matricial artificial dissipation can be used in conjunction with the local time step, as it has been done in the previous section, its use with the block-Jacobi preconditioning provides a discrete operator where the damping of the high frequency errors is enhanced, especially for highly stretched cells. We have seen (figures 5.2a to 5.2d) that when the matricial dissipation is used many eigenvalues lie close to the Fourier space origin, especially for the cases with $\Delta y/\Delta x \ll 1$. This fact is unfortunate since the ability of the smoother to damp errors is



(a) $\alpha = 45°$ (b) $\alpha = \Delta y/\Delta x$ (c) $\alpha = 0$

Figure 5.3: Influence of the angle between the grid and the velocity, α, on the eigenvalue footprint of the spatial discretisation. Matricial artificial dissipation with block-Jacobi preconditioning. $\Delta y/\Delta x \ll 1$ and $M = 0.5$.

very poor in this region, and the multigrid algorithm, that entirely relies on the proper damping of the high frequency errors, looses all its efficiency. Thus, we analyse in detail the case with highly stretched cells, i.e., $\Delta y/\Delta x \ll 1$, which is the one where the enhancement of the damping is more necessary. We are going to distinguish three cases of study, depending on the value of the flow angle α, since the behaviour of the eigenvalues in such cases presents some differences. For all three cases we assume that $M \gg \Delta y/\Delta x$.

$\boldsymbol{\alpha \gg \Delta y/\Delta x}$. For this value of the angle the inviscid parts of the preconditioning matrix and the discrete residual operator yield

$$P_J = \frac{CFL \cdot \vartheta_i}{2\Delta x} |A_y|^{-1} \tag{5.16}$$

and

$$Z = T_y \left[\Lambda_y \cdot i\Delta x \cdot \sin n\Delta y - |\Lambda_y| \left(1 - \cos n\Delta y\right) \cdot \Delta x\right] \cdot T_y^{-1}, \tag{5.17}$$

respectively. The eigenvalues of $P_J \cdot Z/\vartheta_i$ are

$$\frac{CFL}{2} \left\{ \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix} \cdot i \sin n\Delta y - \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \left(1 - \cos n\Delta y\right) \right\}, \tag{5.18}$$

since $|A_y|^{-1} = T_y |\Lambda_y|^{-1} T_y^{-1}$. When comparing these eigenvalues with those of Eqs. (5.14) and (5.15), obtained with the scalar and matricial formulations of the artificial dissipations, we observe that all the high frequency modes of Eq. (5.18) are properly damped and propagated,

except the H_xL_y wave numbers of all the families, that lie close to the real axis (figure 5.3a).

$\alpha = \Delta y/\Delta x$. In this case the eigenvalues of Eq. (5.18) are modified, because the entropy and vorticity eigenvalues of $|A_y|$ are comparable to those of $|A_x| \cdot \Delta y/\Delta x$. The new block-Jacobi preconditioning matrix is

$$P_J^{-1} = \frac{1}{\vartheta_i}\frac{2\Delta x}{CFL}\left(\frac{\Delta y}{\Delta x}|A_x| + |A_y|\right),$$

which provides a new set of eigenvalues of $P_J \cdot Z/\vartheta_i$ [75]:

$$\frac{CFL}{2}\begin{bmatrix} i\frac{1}{2}(\sin m\Delta x + \sin n\Delta y) + \frac{1}{2}(2 - \cos m\Delta x - \cos n\Delta y) \\ i\frac{M}{1+M}(\sin m\Delta x + \sin n\Delta y) + \frac{1}{1+M}(1 - \cos m\Delta x) + \frac{M}{1+M}(1 - \cos n\Delta y) \\ i\sin n\Delta y + (1 - \cos n\Delta y) \\ -i\sin n\Delta y + (1 - \cos n\Delta y) \end{bmatrix} \tag{5.19}$$

The entropy and vorticity eigenvalues of Eq. (5.19) are different from those of Eq. (5.18) in the following sense:

- The eigenvalues of the entropy family are now clustered away from the Fourier space origin for all high frequency wave numbers.
- The eigenvalues of the vorticity family depend on the Mach number. Thus, for low Mach numbers, the phase velocity of the preconditioned vorticity wave tends to zero and the L_xH_y vorticity modes are hardly damped. These effects are corrected with the low Mach number preconditioning.

The footprint of the eigenvalues for this case is depicted in figure 5.3b.

$\alpha \ll \Delta y/\Delta x$. For this case ($\alpha = 0$ is the limit value used in figure 5.3c), the entropy and vorticity eigenvalues are also different, since now $|A_y|$ is singular. The modified eigenvalues of these families are [75]:

$$\frac{CFL}{2}\begin{bmatrix} i\sin m\Delta x + (1 - \cos m\Delta x) \\ iM\sin m\Delta x + (1 - \cos m\Delta x) \end{bmatrix} \tag{5.20}$$

The main difference between these eigenvalues and those of Eq. (5.19) is that the dependence on the $n\Delta y$ modes has been removed when the flow is completely mesh-aligned. Thus, the L_xH_y modes of the vorticity and the entropy waves are now attached to the real axis of the Fourier space, and hence they will be also hard to damp. This case is avoided with the use of the

entropy fix, that limits the minimum eigenvalues of $|A_y|$, hence avoiding its singularity at $v = 0$. This fix also modifies the case with $\alpha = \Delta y/\Delta x \ll 1$. When the limit value ε of Eq. (5.10) is $\varepsilon > |v|$, the eigenvalues of $|\Lambda_y|$ are modified, yielding $\left|\Lambda_y^*\right| = [|v|^* \ |v|^* \ |v+c| \ |v-c|]$, where $|v|^*$ is provided by Eq. (5.10). The expressions of the preconditioning matrix (Eq. (5.16)) and the discrete operator (Eq. (5.17)) are

$$P_J = \frac{CFL \cdot \vartheta_i}{2\Delta x} \left|A_y^*\right|^{-1}$$

and

$$Z = T_y \left[\Lambda_y \cdot i\Delta x \cdot \sin n\Delta y - \left|\Lambda_y^*\right| (1 - \cos n\Delta y) \cdot \Delta x\right] \cdot T_y^{-1}$$

where the effects due to $|A_x|$ have been neglected. The product $P_J \cdot Z/\vartheta_i$ yields the following eigenvalues:

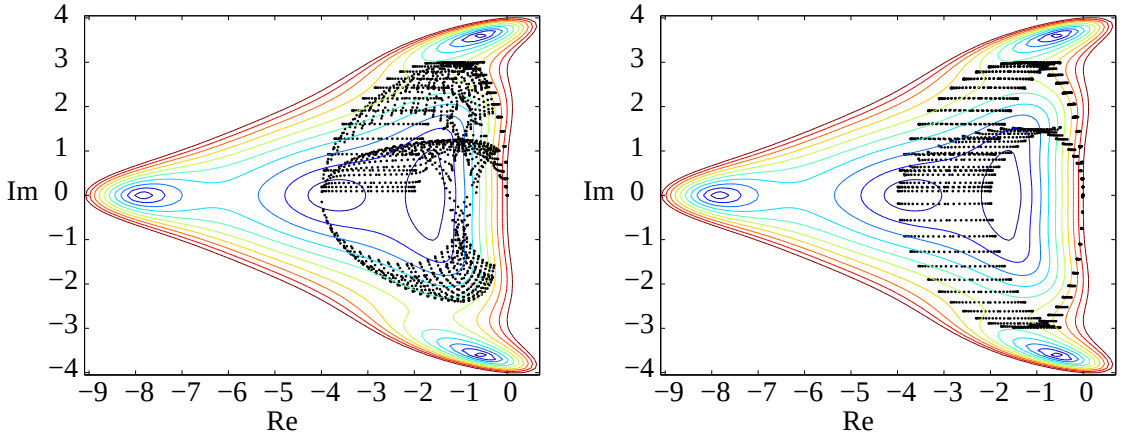
$$\frac{CFL}{2} \left\{ \begin{bmatrix} v/|v|^* \\ v/|v|^* \\ 1 \\ -1 \end{bmatrix} \cdot i \sin n\Delta y - \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} (1 - \cos n\Delta y) \right\}$$

Thus, when $v/|v|^* \ll 1$ the eigenvalues are similar to those of Eq. (5.14). The entropy and vorticity waves are not appropriately propagated, as it was the case with scalar artificial dissipation and local time step.

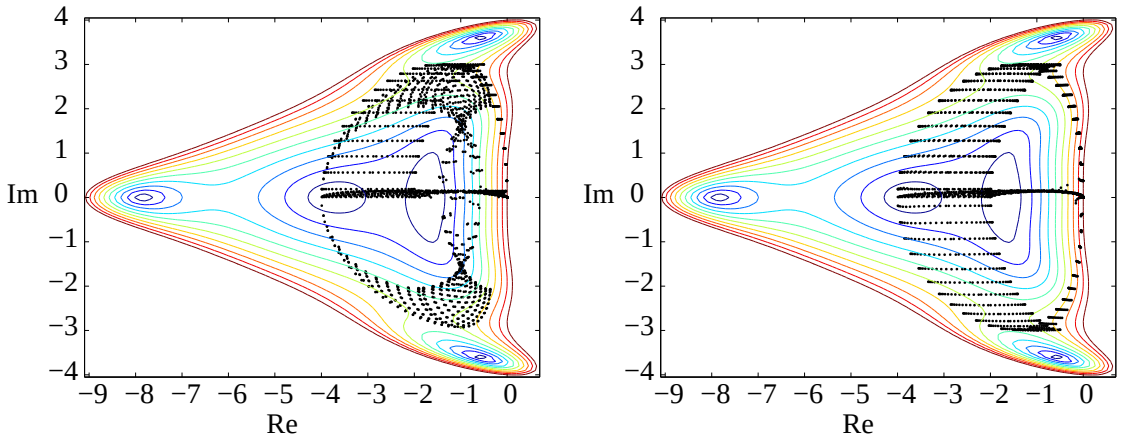
Figure 5.4 shows the footprint of the eigenvalues for the matricial artificial dissipation together with the block-Jacobi preconditioning technique for a value $\alpha = 0$ and the entropy fix disabled.

Figure 5.4a ($M = 0.5$ and isotropic cells) shows that almost all eigenvalues are well clustered away from the origin. Only a small portion of them lie next to zero, but this effect, discussed in the previous section, is produced due to the angle between the velocity and the mesh, that yields a footprint close to the Fourier space origin for $k\Delta x = 0$. If the flow is not mesh-aligned, the damping factors for these modes are improved, as it can be seen in figure 5.5, where the footprint has been plot for two different values of the flow angle while the rest of the parameters remain constant. The Mach number for this case is 0.5 and the cells are isotropic.

Figure 5.4c ($M = 0.05$ and isotropic cells) is similar to 5.1c, but now the low Mach number only provokes that the vorticity eigenvalues lie next to the real axis of the Fourier space for isotropic cells. The entropy eigenvalues are appropriately moved far away from this zone, and hence their propagation is improved. The position of the vorticity eigenvalues is corrected using

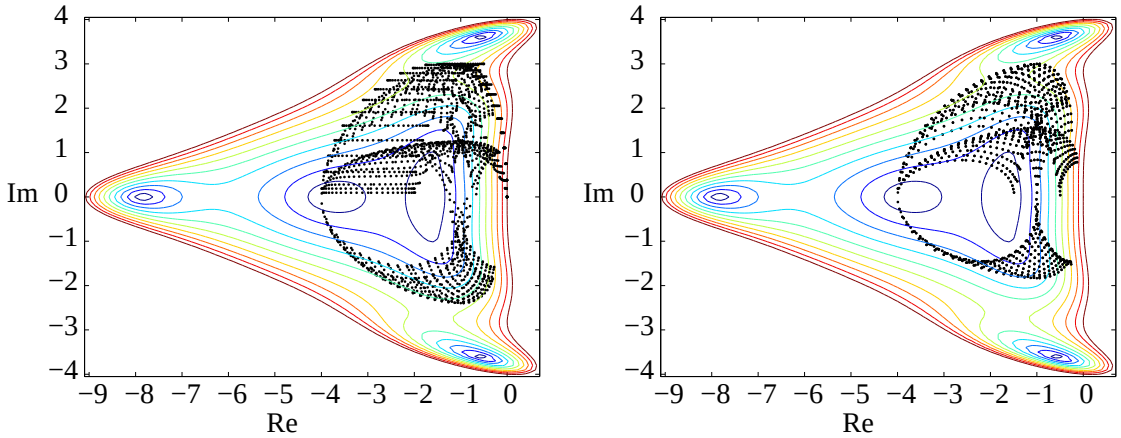


(a) $M = 0.5$, $\Delta y/\Delta x = 1$ (b) $M = 0.5$, $\Delta y/\Delta x \to 0$



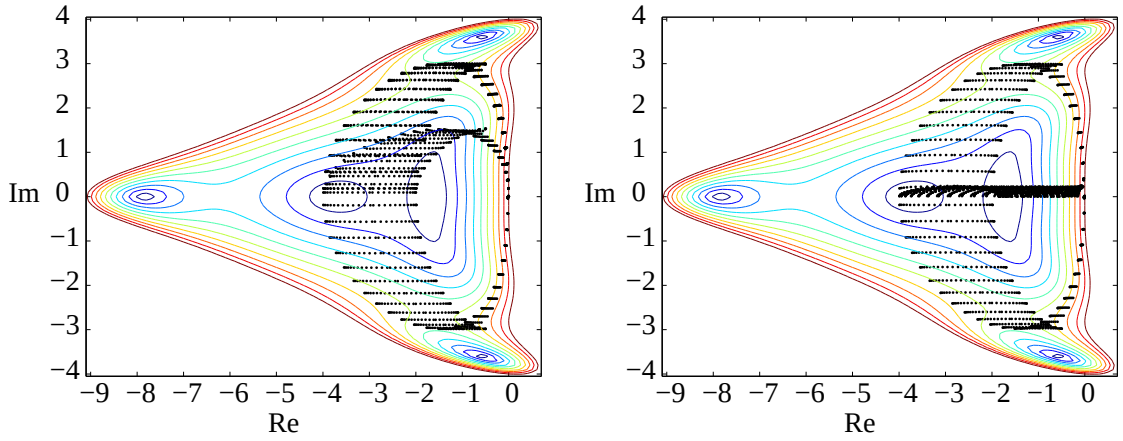
(c) $M = 0.05$, $\Delta y/\Delta x = 1$ (d) $M = 0.05$, $\Delta y/\Delta x \to 0$

Figure 5.4: Influence of the Mach number and grid stretching on the eigenvalue footprint of the spatial discretisation. Matricial artificial dissipation with block-Jacobi preconditioning.



(a) $\alpha = 0°$ (b) $\alpha = 45°$

Figure 5.5: Influence of the angle between the grid and the velocity, α, on the eigenvalue footprint of the spatial discretisation. Matricial artificial dissipation with block-Jacobi preconditioning. $\Delta y/\Delta x = 1$ and $M = 0.5$.



(a) Entropy fix disabled

(b) Entropy fix enabled

Figure 5.6: Influence of the entropy fix on the eigenvalue footprint of the spatial discretisation. Matricial artificial dissipation with block-Jacobi preconditioning. $\Delta y/\Delta x \ll 1$ and $M = 0.5$.

the low Mach number preconditioning. When dealing with stretched cells and $M = 0.5$ (figure 5.4b), we can notice that certainly more eigenvalues are moved away from the Fourier space origin compared to the case with the local time step, but not all them. There is a family of acoustic eigenvalues (the $H_x L_y$ wave numbers) and a family of vorticity and entropy eigenvalues (the $L_x H_y$ wave numbers) that are attached to the imaginary axis and hence they will have poor damping. As long as the preconditioning technique does not move all the eigenvalues further away from the origin, the multigrid technique will remain inefficient.

In this particular case, we have previously mentioned that the entropy fix modifies the shape of the footprints. Figure 5.6 compares the footprint of the eigenvalues when the entropy fix is disabled and enabled, remaining the rest of the parameters constant. The effect of enabling the entropy fix is that the entropy and vorticity waves are hardly propagated, since $v = 0$ and $|v|^* = c/8$. The benefit of the entropy fix is that when it is enabled, the $L_x H_y$ wave numbers of the entropy and vorticity eigenvalues lie away from the Fourier space origin.

Figure 5.4d shows the effects of the low Mach number in the eigenvalues footprint, which have been predicted in Eq. (5.20) and are analogous to those commented before. In the next section the effect of the low Mach number preconditioning is shown, and the differences between the damping of the different wave modes of table 5.1 when combined with the block-Jacobi preconditioning are commented in detail.

5.3. Low Mach number preconditioning

In the three cases previously analysed, we have seen (figures 5.1d, 5.2d and 5.4d) that at low Mach numbers, even with the use of the block-Jacobi preconditioning, there is a group of eigenvalues that are not appropriately convected, i.e.: they are clustered next to the real axis of the Fourier space. This fact does not pose any problem from the point of view of using a multigrid technique to converge the problem, since most of the error modes have enough damping (except when using the matricial artificial dissipation with the local time step -figure 5.2d-). In this case, the stiffness is derived from the different convection speed of the eigenmodes, as it has been commented in the introduction of this chapter. Besides, the excess of damping of certain modes affects the quality of the steady solution. This is the reason of combining the block-Jacobi and the low Mach number preconditioning techniques. Block-Jacobi preconditioning removes the discrete stiffness while low Mach number preconditioning removes the physical one.

Some ideas are first presented in order to understand the formulation of the preconditioning matrix. These ideas are widely reviewed in [83], and their application to unstructured grids can be found in [63]. The low Mach number preconditioning technique modifies the physical eigenvalues, and makes all of them $\mathcal{O}(u)$. To derive the three dimensional formulation of the low Mach number preconditioning matrix we will formulate first the preconditioned one-dimensional linearised Euler equations:

$$\frac{d\mathbf{u}}{dt} + PA\frac{d\mathbf{u}}{dx} = 0$$

where $d\mathbf{u} = [d\rho \quad d(\rho u) \quad d(\rho E)]^T$. The preconditioning matrix takes a particularly simple form if we express the problem in symmetrised variables $d\widetilde{\mathbf{u}} = \left[dp/\rho c \quad du \quad dp - c^2 d\rho\right]^T$, giving

$$\frac{d\widetilde{\mathbf{u}}}{dt} + \Gamma\widetilde{A}\frac{d\widetilde{\mathbf{u}}}{dx} = 0$$

being the preconditioning matrix

$$\Gamma = \begin{bmatrix} \varepsilon & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and

$$\widetilde{A} = \begin{bmatrix} u & c & 0 \\ c & u & 0 \\ 0 & 0 & u \end{bmatrix}$$

Taking $\varepsilon \propto M^2$ ensures that the acoustic wave speeds are similar to those of convective waves, which are proportional to u. When $\varepsilon = 1$ the unpreconditioned formulation is recovered, which is the case if the Mach number is high enough (by construction of the ε parameter). The determination of ε is not obvious for internal flows [63], and will not be addressed here, while for external flows, $\varepsilon = k \cdot M_\infty^2$, being k a constant, and M_∞ the Mach number of the free stream.

Expressing the system in semi-discrete formulation, and using a first order upwind discretisation, we obtain

$$\frac{d\widetilde{\mathbf{u}}_i}{dt} + \Gamma\widetilde{A}\frac{\widetilde{\mathbf{u}}_{i+1} - \widetilde{\mathbf{u}}_{i-1}}{2\Delta x} - \left|\Gamma\widetilde{A}\right| \frac{\widetilde{\mathbf{u}}_{i+1} - 2\widetilde{\mathbf{u}}_i + \widetilde{\mathbf{u}}_{i-1}}{2\Delta x} = 0,$$

which, premultiplying by Γ^{-1} and turning back to conservative linearised variables,

$$\widetilde{M}\Gamma^{-1}\widetilde{M}^{-1}\frac{d\mathbf{u}}{dt} + A\frac{\mathbf{u}_{i+1} - \mathbf{u}_{i-1}}{2\Delta x} - \widetilde{M}\Gamma^{-1}\left|\Gamma\widetilde{A}\right|\widetilde{M}^{-1}\frac{\mathbf{u}_{i+1} - 2\mathbf{u}_i + \mathbf{u}_{i-1}}{2\Delta x} = 0 \tag{5.21}$$

where $\widetilde{M} = \partial\mathbf{u}/\partial\widetilde{\mathbf{u}}$ is the transformation matrix. This equation yields a modified artificial dissipation matrix with the changes introduced by the low Mach preconditioning matrix Γ.

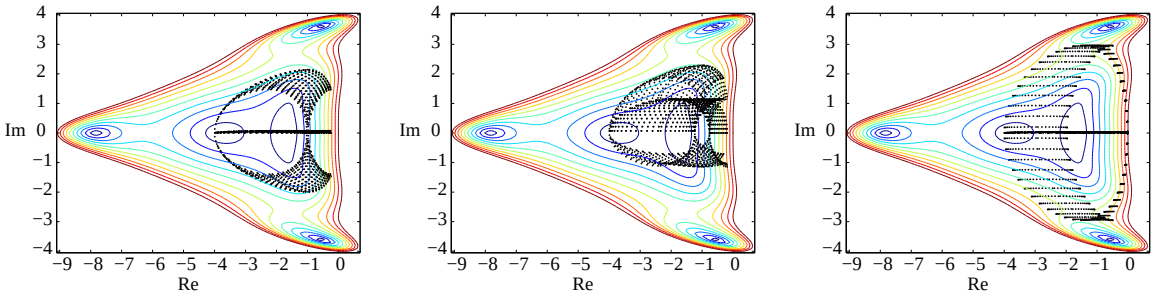
Extending equation (5.21) to three dimensional unstructured grids is simple. We just need to rewrite equation (5.2) taking into account the modification in the artificial dissipation terms which is derived from equation (5.21):

$$\begin{aligned}\mathbf{F}_{ij}^I &= \frac{1}{2}\left[A_{ij}\left(\mathbf{u}_i + \mathbf{u}_j\right) - \right.\\ &- \widetilde{M}_{ij}\Gamma_{ij}^{-1}\left|\Gamma_{ij}\widetilde{A}_{ij}\right|\widetilde{M}_{ij}^{-1}\left(-\frac{1}{2}\left(1-\kappa\right)\left(1-S\right)\left(L_i\left(\mathbf{u}\right) - L_j\left(\mathbf{u}\right)\right) + \right.\\ &+ \left.\left. S\left(\mathbf{u}_i - \mathbf{u}_j\right)\right)\right]\end{aligned} \tag{5.22}$$

where

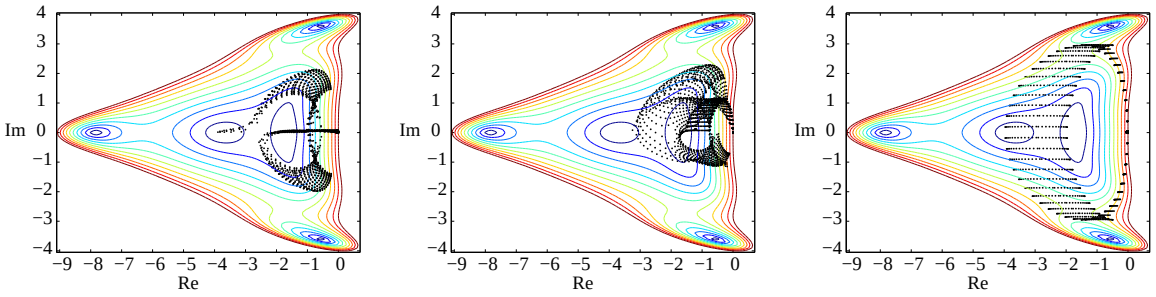
$$\Gamma_{ij} = \begin{bmatrix} \varepsilon_{ij} & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Once we have obtained this new formulation, the modified block-Jacobi preconditioning matrix is obtained by extracting the terms that affect the central node of the discretisation. As it was done when constructing the block-Jacobi preconditioning in section 5.1.2, only the second



(a) $\Delta y/\Delta x = 1$ (b) $\Delta y/\Delta x = 1$ (c) $\Delta y/\Delta x \to 0$

Figure 5.7: Scalar artificial dissipation with local time step for $M = 0.05$. (a) without low Mach number preconditioning, (b) and (c) with low Mach number preconditioning



(a) $\Delta y/\Delta x = 1$ (b) $\Delta y/\Delta x = 1$ (c) $\Delta y/\Delta x \to 0$

Figure 5.8: Matricial artificial dissipation with local time step for $M = 0.05$. (a) without low Mach number preconditioning, (b) and (c) with low Mach number preconditioning

differences term is taken into account:

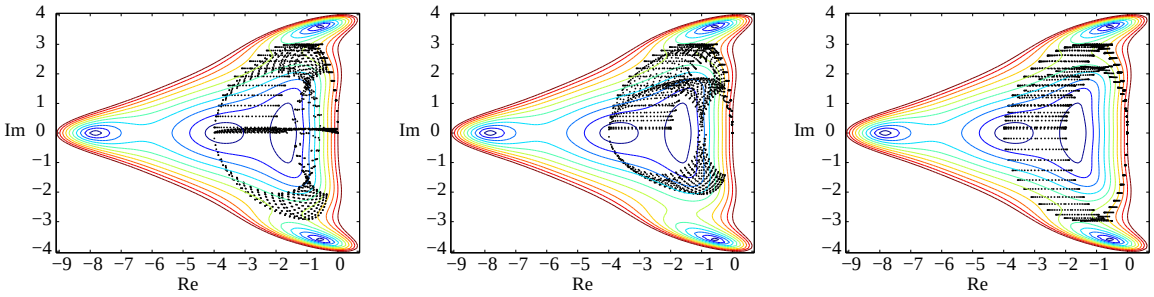
$$P^{-1}_{LM-J_i} = \frac{1}{V_i}\frac{1}{CFL_H}\left[\sum_{j=1}^{\#ed_i}\left(\frac{1}{2}\widetilde{M}_{ij}\Gamma^{-1}_{ij}\left|\Gamma_{ij}\widetilde{A}_{ij}\right|\widetilde{M}^{-1}_{ij} + B_{ij}\frac{1}{|\mathbf{x}_j - \mathbf{x}_i|}\right)\sigma_{ij} + \sum_{j=1}^{\#Bed_i}\frac{1}{2}\widetilde{M}_{ij}\Gamma^{-1}_{ij}\left|\Gamma_{ij}\widetilde{A}_{ij}\right|\widetilde{M}^{-1}_{ij}\sigma_{ij}\right] \tag{5.23}$$

The matrices involved in the low Mach number preconditioning are written in appendix A.

5.3.1. Stability analysis

The analysis of the low Mach number preconditioning is analogous to the one presented in section 5.2. The residual expressed in equation (5.11) is used, but the matrix $|A_{ij}|$ has been substituted by the modified artificial dissipation matrix, $\widetilde{M}_{ij}\Gamma^{-1}_{ij}\left|\Gamma_{ij}\widetilde{A}_{ij}\right|\widetilde{M}^{-1}_{ij}$.

The effects of this preconditioning technique are seen in figures 5.7, 5.8 and 5.9, where three cases are represented, all of them with $M = 0.05$ and with the rest of significant parameters having an identical value to those used in section 5.2. The left column represents the Fourier footprint of the eigenvalues associated to the error modes of table 5.1 without using the low

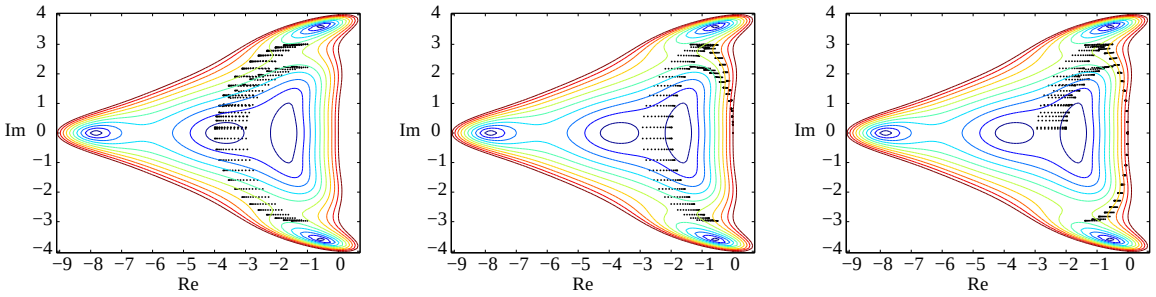


(a) $\Delta y/\Delta x = 1$ (b) $\Delta y/\Delta x = 1$ (c) $\Delta y/\Delta x \to 0$

Figure 5.9: Matricial artificial dissipation with block-Jacobi preconditioning for $M = 0.05$. (a) without low Mach number preconditioning, (b) and (c) with low Mach number preconditioning

Mach number preconditioning technique for an isotropic mesh. The central column shows the effects of using the low Mach number preconditioning in the same mesh and the right column shows the effect of the grid stretching in the low Mach number preconditioned residual. The differences between the Fourier footprints for the cases without stretching, with and without low Mach number preconditioning (columns (a) and (b)), are identical for the three cases represented. Thus, in figures 5.7a, 5.8a and 5.9a, it is seen that a group of error modes lies in the real axis of the Fourier space, meaning that these errors are just damped but not propagated. This is produced by the physical stiffness associated with the low Mach number regime. The numerical effects are twofold: first the convergence rate is degraded, since the error modes are hardly propagated and not easily expelled out of the domain and second, an excess of dissipation is produced, hence degrading the quality of the steady solution, in a similar manner to that expounded in the section 5.1 for the scalar artificial dissipation model. These effects are corrected when using the low Mach number preconditioning technique (figures 5.7b, 5.8b and 5.9b). In all these figures it is observed that the use of the low Mach number preconditioning prevents the effects commented before, i.e.: the eigenvalues are moved away from the real axis, hence the source of stiffness is removed. Even though the eigenvalues footprint depicted in the above mentioned figures stands just for the high frequency error modes, the situation is analogous for the low frequency modes, therefore the actions taken to correct the overdamping of the oscillatory modes also correct the smooth ones, which actually are the modes that have to be correctly solved.

When we are dealing with stretched cells, the same problems previously described are reproduced, since the low Mach number preconditioning alone can not fix the problems associated with the stretched cells. Thus, figures 5.7c, 5.8c, corresponding to the low Mach number formulation of the scalar and matricial artificial viscosity models with the local time step for stretched



(a) H_xH_y wave numbers (b) L_xH_y wave numbers (c) H_xL_y wave numbers

Figure 5.10: Matricial artificial dissipation with block-Jacobi and low Mach preconditioning for $M = 0.05$

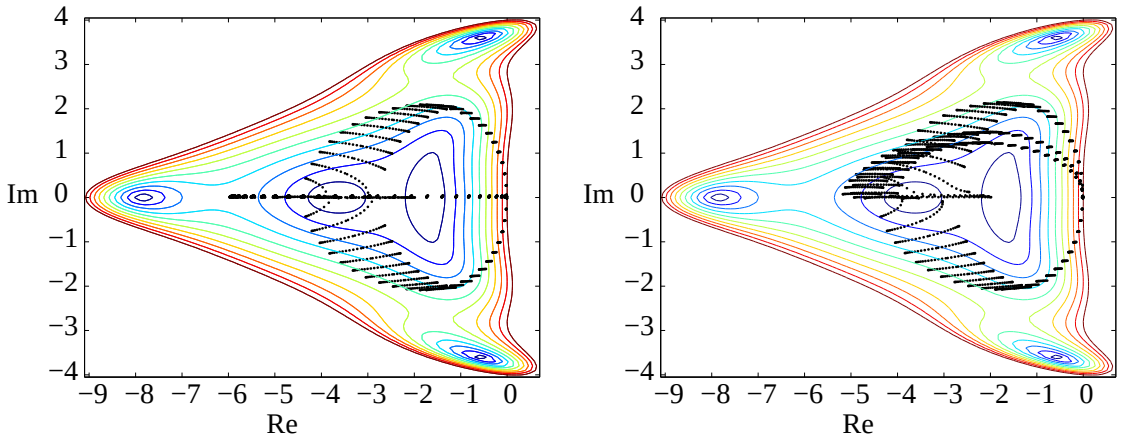
cells, are essentially equal to those where the low Mach number preconditioning has not been used (figures 5.1d and 5.2d). On the other hand, figures 5.4d and 5.9c show different behaviours. While the former has some eigenvalues attached to the Fourier space real axis, the latter succeeds in moving them further from that zone, hence relieving the effects of the low Mach number over the convergence rate. As it was expected, the use of the combined block-Jacobi and low Mach number preconditioning techniques removes the two main source of stiffness commented in the introduction.

It is also seen in figure 5.9c that some high frequency eigenvalues are still clustered in zones of the Fourier space where the damping values of the Runge-Kutta scheme are very low. If we plot each high frequency quadrant of table 5.1 separately, it is possible to appreciate which modes are still loosely damped even with the use of the low Mach and block-Jacobi preconditioning techniques. Figure 5.10a shows the eigenvalues footprint just for the H_xH_y wave numbers. The use of the preconditioning technique succeeds in moving these eigenvalues further away from the Fourier space origin. When the L_xH_y wave numbers are considered (figure 5.10b), the preconditioning matrix has a reasonable success in moving the eigenvalues footprint. However, a few wave numbers corresponding to the entropy and vorticity eigenvalues are still attached next to the imaginary axis, where they are hardly damped. This case is avoided with the use of the entropy fix, as it has been commented in section 5.2.3.The situation worsens when trying to modify the footprint of the H_xL_y wave numbers (figure 5.10c). Even though the block-Jacobi preconditioning technique effectively moves some eigenvalues away from the origin, certainly more than the scalar preconditioning, some eigenvalues that correspond to the acoustic waves lie very close to the origin. It should be noted that this situation cannot be avoided with the use of the entropy fix, like it was done to correct the L_xH_y modes of the entropy and vorticity waves, because the eigenvalues $|u + c|$ and $|u - c|$ will always be greater than the limit of the entropy

fix. Therefore, if we are using a multigrid method, these error modes will not be appropriately damped and the solution will not be smooth enough to be solved in a coarser multigrid level. If we coarsen the fine grid just in the direction where the preconditioning technique is able to damp high frequency, then all grids solve smooth errors and the loss of efficiency experimented by the multigrid is avoided. This is accomplished by semi-coarsening just in the wall normal direction, and leaving the streamwise direction unaltered. This semi-coarsening is done just in the stretched cells, and the memory and CPU time overhead are not greatly increased with respect to the full coarsening case (see [60]).

5.4. Influence of the viscous terms

All the analyses performed until this point have neglected the contribution of the viscous terms in the different preconditioning techniques and the residual of equation (5.11), by making the assumption that the cell Reynolds number, $Re_{\Delta x}$, was large enough. However, when the cells are small enough, this assumption is not true anymore. In that case, the footprints of the eigenvalues are modified. Figure 5.11a illustrates such modifications, when $Re_{\Delta x} \sim \mathcal{O}(100)$ and the aspect ratio of the cells is large enough to obtain a $Re_{\Delta y} \sim \mathcal{O}(1)$. We can see that many of the eigenvalues are concentrated next to the real axis. However, this phenomenon seems to be strongly dependent on the angle α. Thus, for larger values of this parameter ($\alpha = 45°$), many of the eigenvalues are moved further away from $Im = 0$ (figure 5.11b). Despite this behaviour, the characteristics of the footprint have not been substantially altered, if we compare them with those of figure 5.10c. We also observe that the majority of the eigenvalues have been moved away from the origin, and some of them lie close to the imaginary axis, where the damping is low.



(a) $\alpha = 0°$ (b) $\alpha = 45°$

Figure 5.11: Influence of the viscous terms in the eigenvalues footprint. Matricial artificial dissipation with block-Jacobi and low Mach preconditioning for $M = 0.05$, $\Delta y/\Delta x \to 0$

6. Dual Time Step

The analysis of unsteady problems requires a fixed global time step to accurately simulate unsteady phenomena. This fact does not allow the use of local time stepping and other convergence acceleration techniques (preconditioning, multigrid, residual smoothing, etc.). In problems where the grid has elements of varying size, the stability criterion in the smallest grid volume fixes a maximum allowable time step for the whole domain. Usually this time step is so small that running an unsteady case (e.g.; a vibrating blade or a rotor-stator interaction) becomes prohibitively expensive when explicit algorithms are used to march in time. For example, solving the periodic vibration of a blade with a vibration frequency of 200 Hz with a typical mesh for a state of the art design simulation, which may have a maximum allowable time step of $\mathcal{O}(10^{-7})$ seconds, would require about 50000 time steps per period, which is too costly (especially if 5-10 periods are needed to converge to a periodic state).

This problem is solved using implicit algorithms to march in time. Since these methods do not limit the maximum time step, it can be chosen to meet temporal accuracy requirements, avoiding stability restrictions. However, classical implicit methods require a larger amount of memory and are also hard to parallelise. The framework constructed to efficiently solve steady-state problems may be re-used to construct the so-called dual time step method. This approach was first proposed by Jameson [43], and since then it has been widely used in the resolution of non-linear unsteady problems [2, 5].

6.1. Dual Time Step Equations

We start writing the semi-discrete form of the Navier-Stokes equations:

$$\frac{d\mathbf{U}}{dt} + R\left(\mathbf{U}\right) = 0, \tag{6.1}$$

As presented in previous chapters, several techniques are available to accelerate the convergence to the steady state, $R(\mathbf{U}) = 0$. However, in unsteady problems the system of equations (6.1) needs to be solved in a time-accurate manner. Explicit time marching algorithms have sometimes very stringent stability restrictions and in these cases implicit methods are preferred to solve the system. This situation is presented when the time scale we are interested in is much larger that the shortest time scale that is solved in the simulation. This is typically the case when the stability restrictions that arise due to the spatial accuracy requirements impose a time step much shorter than the one that would be selected based on temporal accuracy requirements.

The dual time step method introduces an additional term in equation (6.1), consisting in the derivative of the conservative variables with respect to a fictitious or dual time τ. The resulting equation is:

$$\frac{\partial \mathbf{U}^*}{\partial \tau} + \left[\frac{\partial \mathbf{U}^*}{\partial t} + R(\mathbf{U}^*) \right] = 0 \tag{6.2}$$

This equation can be solved using steady state acceleration techniques, marching in τ instead of t. When this new problem is converged to a steady state, Eq. (6.1) is recovered. Thus, the problem has been reduced to achieve the solution of a steady problem with a time flux added. The discretisation for this physical time flux was also proposed by Jameson [43], a three-point backward formula, second-order accurate in time:

$$\frac{\partial (\mathbf{U}^*)^{n+1}}{\partial t} = \frac{3(\mathbf{U}^*)^{n+1} - 4(\mathbf{U}^*)^n + (\mathbf{U}^*)^{n-1}}{2\Delta t} \tag{6.3}$$

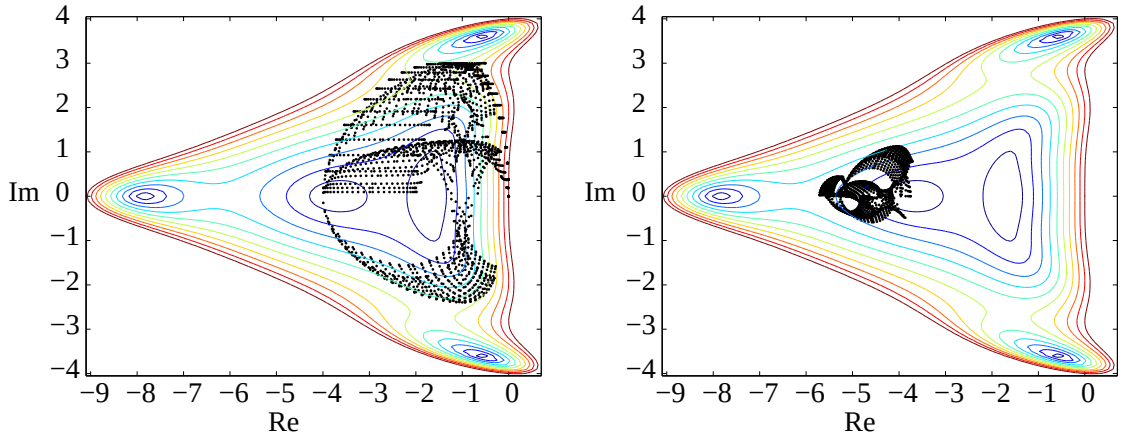
The full semi-discrete scheme is then written as:

$$\frac{d(\mathbf{U}^*)^{n+1}}{d\tau} + \frac{3(\mathbf{U}^*)^{n+1} - 4(\mathbf{U}^*)^n + (\mathbf{U}^*)^{n-1}}{2\Delta t} + R\left((\mathbf{U}^*)^{n+1}\right) = 0 \tag{6.4}$$

that in compact form may be written as

$$\frac{d(\mathbf{U}^*)^{n+1}}{d\tau} + R^*\left((\mathbf{U}^*)^{n+1}\right) = 0$$

The superscript n is refers to the physical time step, thus when the steady state of the problem is reached, what we have actually obtained is the solution of the unsteady problem in $n+1$. The discretisation written in equation (6.4) is implicit when marching in the physical time but explicit when marching in the dual one.



(a) (b)

Figure 6.1: Matricial dissipation with block-Jacobi preconditioning. (a) $\Delta t \gg |\mathbf{x}_j - \mathbf{x}_i| / |\mathbf{u}_i|$ **(b)** $\Delta t \sim \mathcal{O}(|\mathbf{x}_j - \mathbf{x}_i| / |\mathbf{u}_i|)$

6.2. Stability analysis

The inclusion of the time derivative in the residual R adds an additional term to the fluxes expression of Eq. (2.7), and hence the expression of the discrete spatial operator (Eq. (5.11)) is also modified by adding the term that affects the central node of the stencil, since the rest of the terms of equation (6.3) remain constant during the convergence of each time step:

$$Z_{DTS} = Z_{NonDTS} - \frac{3}{2}\frac{\vartheta_i}{\Delta t} \cdot I \tag{6.5}$$

where Z_{NonDTS} is written in equation (5.11). The residual modification demands changes in the expressions of both the local time step and the block-Jacobi preconditioning matrix. The modification is analogous to that introduced in equation (6.5):

$$P_{Ji}^{-1}\big|_{DTS} = P_{Ji}^{-1}\big|_{NonDTS} + \frac{1}{CFL_H}\frac{3}{2\Delta t} \cdot I \tag{6.6}$$

where $P_{Ji}^{-1}\big|_{DTS}$ is given by equation (5.9) and I is the 5×5 identity matrix. The same changes should be made in the expression of the local time step of Eq. (5.6), but just using the scalar that precedes the identity matrix in the previous equation.

The additional term of equations (6.5) and (6.6) does not affect very much the eigenvalues of the $P \cdot Z$ operator that have been analysed in chapter 5 when $\Delta t \gg \Delta\tau$, but for small values of the time step the eigenvalues are strongly modified. In the limit $\Delta t/\Delta\tau \ll 1$, the dual time step term is dominant in both equations (6.5) and (6.6), and the $P \cdot Z/\vartheta_i$ product is reduced to the

scalar value $-CFL_H$, therefore all eigenvalues tend to concentrate to that point in the real axis of the Fourier space, as it is shown in figure 6.1b, where the eigenvalues footprint of a case with dual time step are plotted. The parameters used to obtain the eigenvalue footprint are those of section 5.2. This figure also shows that when using the dual time step with a small physical time step, of the order of the characteristic time of the cell $|\mathbf{x}_j - \mathbf{x}_i| \,/\, |\mathbf{u}_i|$, the eigenvalues are concentrated in an area with very high damping and hence the high frequency errors are very well damped. Therefore, all the techniques employed to enhance the damping of such error modes are not necessary for this special case. However, it must be noticed that the purpose of the dual time step is to avoid having to run with such small time steps, and hence the situation of figure 6.1b is rare, except for cases with very large cell size disparity, where the largest cells could fall into this situation.

7. Results

This chapter presents some applications of the convergence acceleration techniques for the resolution of the Euler and RANS equations in realistic configurations. The solver, known as Mu^2s^2T, has been used at ITP for the design of several low pressure turbines. The degree of sophistication of the numerical algorithms has evolved with the pass of time, and the last applications feature all the numerical characteristics described in this thesis, together with the parallelisation of the solver, which has been carried using MPI [19]. However, we have preferred to keep the examples presented here within a more academic environment, showing the result in simplified configurations that may be referred in the open literature. We are presenting results for the T106 linear cascade [87], and also for a low pressure turbine row designed at ITP. We have included as well airfoil vibration cases to illustrate the effectiveness of the dual time step.

7.1. T106 linear cascade

The boundary conditions for all the two-dimensional simulations of this case are shown in table 7.1. This airfoil has been simulated in both two and three dimensions. In the two-dimensional case, we have performed inviscid and viscous simulations, using the Baldwin-Lomax turbulence model as a closure [7] in the latter case. Even though the $k-\omega$ turbulence model [86] is also implemented in the code, its coupling with the main equations may spoil the whole resolution algorithm, and the grid sensitivity effects that we are trying to isolate are not so clearly seen. For the three-dimensional simulation, the $k-\omega$ turbulence model has been used, since it is believed to be a more consistent framework to represent secondary flows.

p_{t1}	$101300Pa$
T_{t1}	$578.6K$
α_1	-37.7°
p_2	$80039.4Pa$

Table 7.1: Flow conditions for T106 case

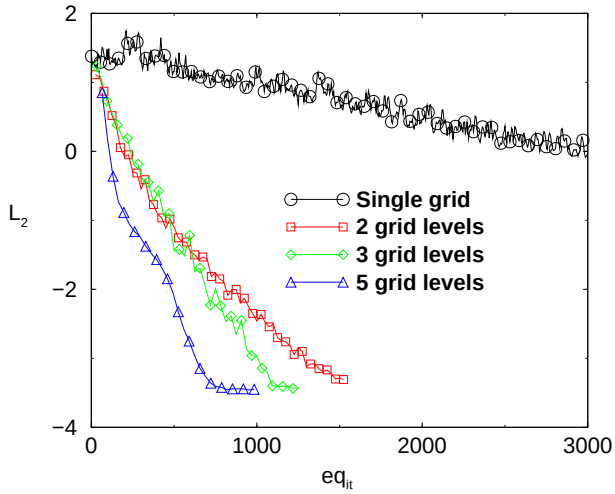


Figure 7.1: Influence of the number of multigrid levels on the convergence rate of a two-dimensional inviscid simulation of the T106 airfoil using the Roe's model for the artificial viscosity and block-Jacobi preconditioning. Multigrid V-cycle. 48400 nodes.

7.1.1. Euler simulations

Euler grids are, in first approximation, quasi-isotropic grids, since the strong gradients that exist in the boundary layers are not present. In inviscid cases, strong gradients are produced due to the presence of shock waves in transonic cases and also in the vicinity of the leading edge of the airfoils. In these zones, the grid is locally refined to capture the smaller scales, but always using isotropic cells. We have seen in section 5.2 that this isotropy favours the convergence of the multigrid scheme. The use of the multigrid method yields a convergence rate which is independent from the cell size and hence a large increase in the convergence rate is expected. This idea will be proved simulating the same geometry in two different grids, the first having four times more points than the second, and showing that convergence rate depends only on the number of nodes in the coarse grid. The isotropic grids that have been used are two-dimensional: the fine grid has 190753 nodes and the coarse one has 48400.

Figure 7.1 depicts the influence of the number of multigrid levels on the convergence rate for the 48400 point grid. A sequence of 4 grids, each coarser that the previous, has been generated, having 13855, 4017, 1214 and 389 nodes respectively. The equivalent iterations used to compare the single grid and multigrid results are computed taking into account the edge ratio between the fine and coarse grids. Thus, assuming a constant number of iterations per grid level and an arbitrary edge ratio for each grid level, the cost of a multigrid V-cycle is, according to Eq.

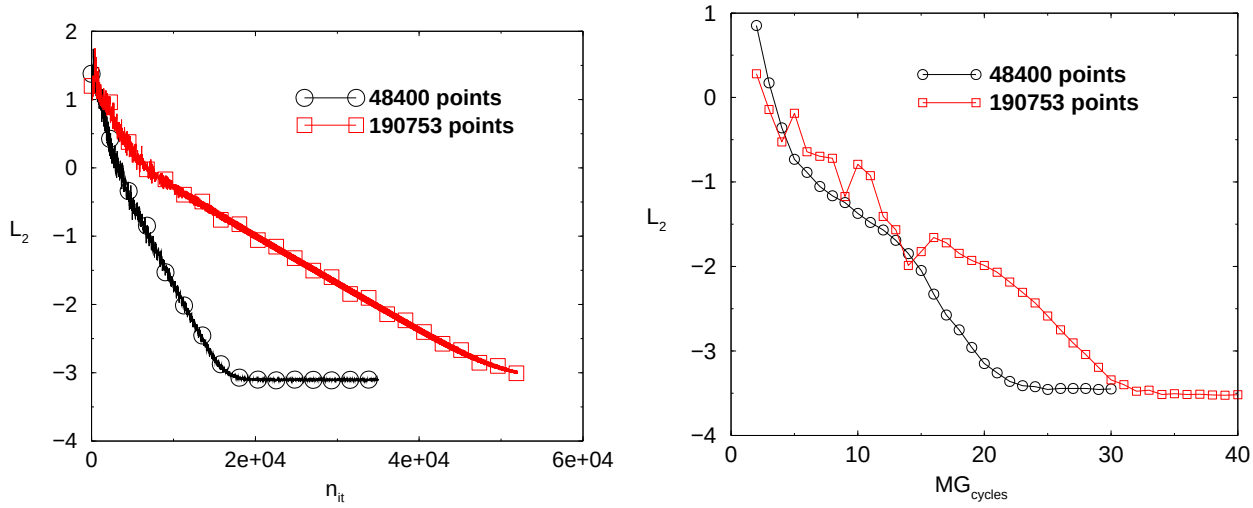


Figure 7.2: Convergence history of a two-dimensional inviscid simulation of the T106 airfoil using the Roe's model for the artificial viscosity and block-Jacobi preconditioning. Left: Single grid method using two different grids. Right: Multigrid V-cycle.

(3.21)

$$C_{MG} = N_{iterations/level} \cdot \left(1 + 2 \left(\sum_{i=2}^{i=MG_{lev}-1} \frac{N_{edge}\,(i)}{N_{edge}\,(i-1)} \right) + \frac{N_{edge}\,(MG_{lev})}{N_{edge}\,(MG_{lev}-1)} \right) \tag{7.1}$$

being $N_{iterations/level}$ the number of iterations per grid level, which has been set to 20 in the present work, and $N_{edge}\,(i)$ the number of edges of the grid level i. Making use of Eq. (7.1), the equivalent iterations used in figure 7.1 is $eq_{it} = N_{MGcycles} \cdot C_{MG}$, being $N_{MGcycles}$ the number of multigrid cycles. A factor of two in the convergence rate is obtained when comparing the convergence with two grid levels and with five grid levels. The convergence rate of the single grid case has been enhanced by a factor of 25 when compared with the five grid levels case. The convergence of the latter to the steady solution requires about 15 minutes of CPU time on a Pentium IV at 1.8 GHz.

In figure 7.2 we show that by using the multigrid method, the convergence rate is (almost) independent from the grid used to discretise the flow domain. The figure on the left shows that by refining the coarse grid, inserting additional points in the middle point of every coarser grid edge, the number of iterations needed to converge the resulting grid is increased by a factor of three. But when the multigrid is enabled, the number of multigrid cycles needed to attain convergence is only a 25% higher. For the finer grid, an additional grid level has been constructed to obtain a number of nodes in the coarser grid level similar to the fifth grid level of the 48400 points grid. For this case, the sequence of the number of nodes for each grid level is: 190753, 55568, 15864, 4572, 1371 and 431.

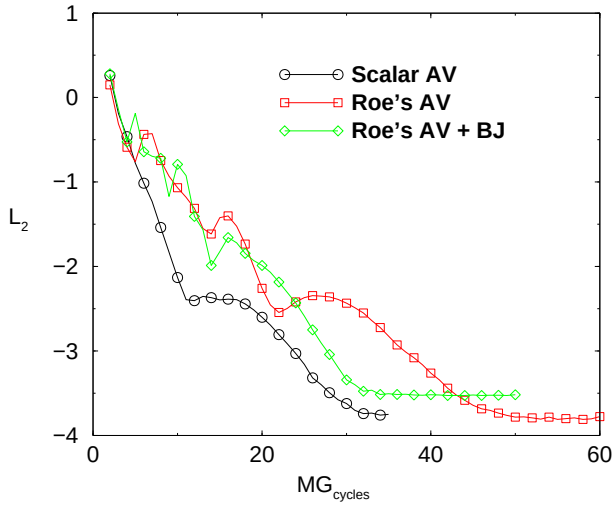


Figure 7.3: Influence of the artificial viscosity model in the convergence rate of a two-dimensional inviscid simulation of the T106 airfoil using a multigrid V-cycle with six grid levels. 190753 nodes.

Figure 7.3 shows that for isotropic grids, the convergence rate becomes independent of the artificial viscosity model used. This result was advanced in section 5.2, where it was demonstrated that for isotropic grids the high frequency error damping is satisfactory whatever the model of artificial viscosity is. Hence, small differences are expected when comparing the scalar artificial viscosity, the matricial Roe's artificial viscosity, and the latter in conjunction with the block-Jacobi preconditioning. This result is corroborated in figure 7.3.

7.1.2. Reynolds-Averaged Navier-Stokes simulations

The main difference between the domain discretisation used for Euler and RANS simulations is that while the former does not have strong gradients next to the walls and uses isotropic grids, the latter also solves the boundary layer region using stretched grids (see figure 7.4). This stretching is a source of stiffness in the discrete equations, and should be accounted for if appropriate convergence rates are to be obtained.

Figure 7.5 shows the reduction of the convergence rate due to the stretching and how the block-Jacobi preconditioning fixes that problem. The example is a RANS simulation of the T106 airfoil using the mixing length turbulence model as a closure and with a grid, depicted in figure 7.4, that has a maximum aspect ratio of 155 in the boundary layer region. The mixing length turbulent model is preferred for this study since its simplicity prevents an interaction with the rest of equations and permits the isolation of the stiffness phenomenon. In the convergence history we can see a first zone, where the convergence rates are very similar in both cases. This stage corresponds to the convergence of the core flow where the cells are not stretched. The

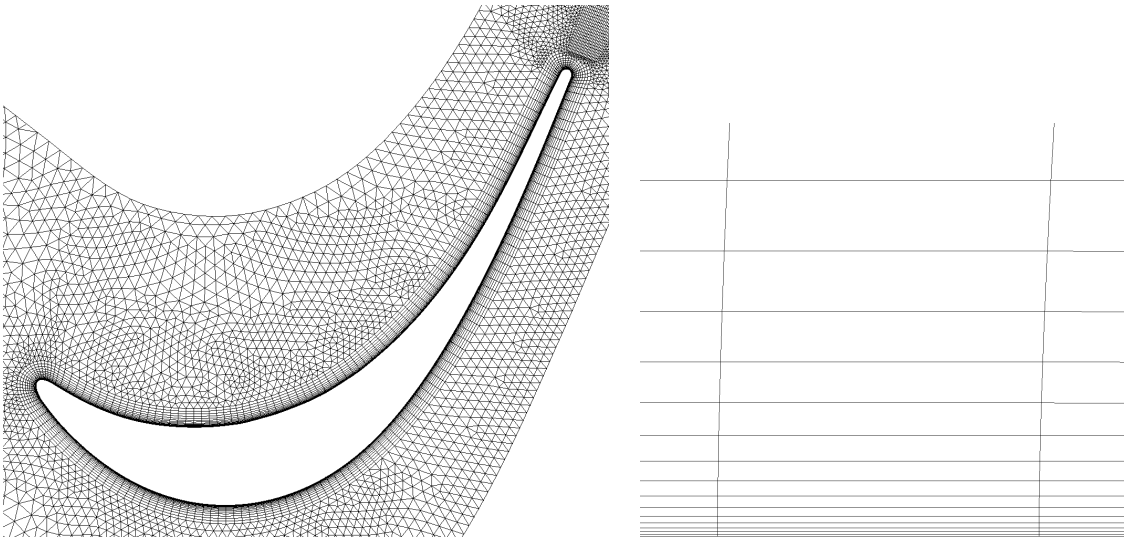


Figure 7.4: Detail of the discretisation of the T106 airfoil for the resolution of the RANS equations, 11641 nodes. Left: view of the core flow and boundary layer grids. Right: detail of the stretched cells in the boundary layer.

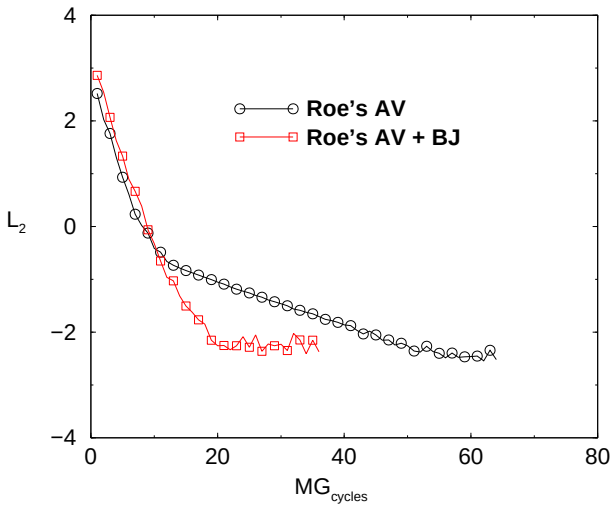


Figure 7.5: Influence of the block-Jacobi preconditioning in the convergence rate of a two-dimensional RANS simulation of the T106 airfoil using the Roe's model for the artificial viscosity and the mixing length turbulence model. 11641 nodes, maximum aspect ratio of 155, multigrid V-cycle with 3 grid levels.

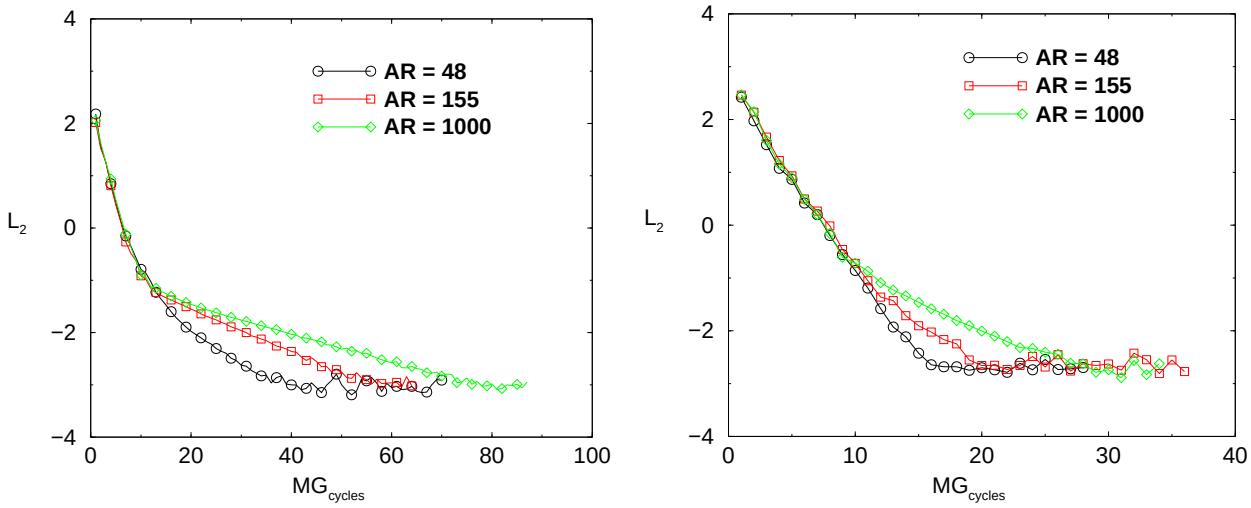


Figure 7.6: Influence of the maximum aspect ratio in the convergence rate of a two-dimensional RANS simulation of the T106 airfoil using the mixing length turbulence model. Left: Roe's artificial viscosity. Right: Roe's artificial viscosity with block-Jacobi preconditioning. 11641 nodes, Multigrid V-cycle with 3 grid levels.

second zone, which corresponds to the full convergence of the boundary layer cells, presents a quite different behaviour. While the convergence rate of the matricial artificial viscosity alone is severely degraded, the use of the block-Jacobi preconditioning maintains the convergence rate of the first region. The number of iterations to reach the steady state is reduced by a factor of three. The reduction in CPU time is lower, since the use of the block-Jacobi preconditioning increases the CPU cost per iteration by about a 20%.

Figure 7.6 shows the convergence history of three cases, each one with the same inviscid grid but with different stretching levels in the boundary layer region, hence modifying the value of the maximum aspect ratio. The left figure displays the convergence histories when the baseline code with matricial artificial viscosity is used, while the right one shows the same cases when the block-Jacobi preconditioning is enabled. It is shown that when there are high frequency error modes that are not properly damped by the smoother, the number of iterations to reach the steady state is no longer independent of the grid. Thus, when the matricial artificial viscosity is used without the block-Jacobi preconditioning, the convergence rate has strong variations. Although these variations also occur when the preconditioning is enabled, the convergence rate is less degraded, even for very high aspect ratios. The variations in the convergence rate could be attributed to the approximation made for the viscous contribution to the preconditioning matrix (see Eq. (5.9)). When these terms are dominant, as it it the case for very low values of Δy and hence for very high aspect ratios, the contribution of the viscous terms to the preconditioning matrix modifies the eigenvalues footprint, pushing some of them into the real axis of the Fourier

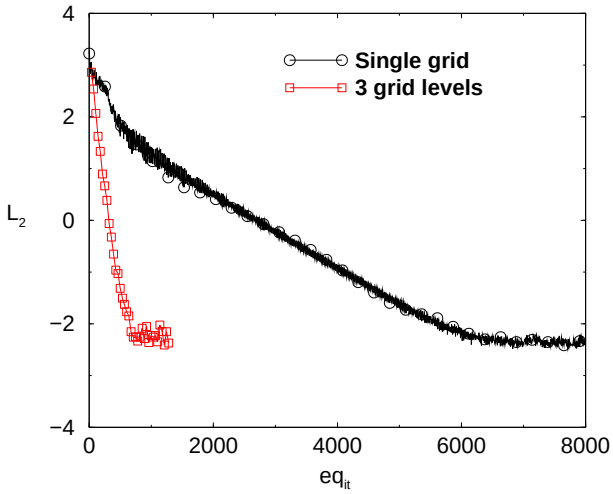


Figure 7.7: Influence of the multigrid scheme on the convergence rate of a two-dimensional RANS simulation of the T106 airfoil using the Roe's model for the artificial viscosity, block-Jacobi preconditioning and the mixing length turbulence model. 11641 nodes, maximum aspect ratio of 155, multigrid V-cycle.

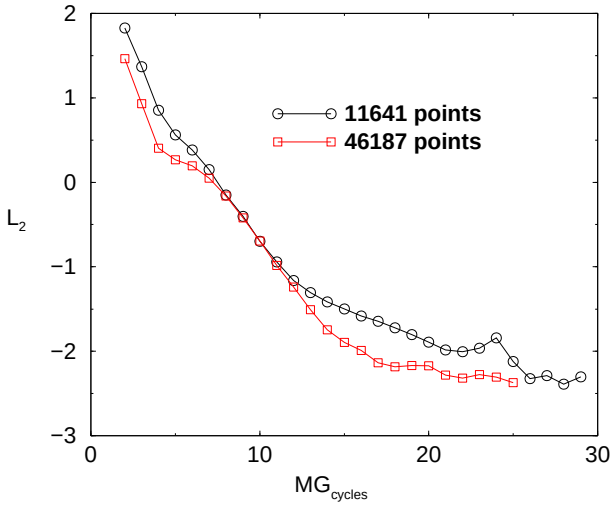


Figure 7.8: Influence of the number of grid points on the convergence rate of a two-dimensional RANS simulation of the T106 airfoil using the Roe's model for the artificial viscosity, block-Jacobi preconditioning and the mixing length turbulence model. Maximum aspect ratio of 155, multigrid FMG V-cycle.

space (see figure 5.11) . Then another source of stiffness is presented, since the propagation speed of the modes that lie next to the real axis almost vanishes, and that source of stiffness is not correctly addressed by the block-Jacobi preconditioning.

The benefits of the multigrid scheme for viscous cases are seen in figure 7.7, where the convergence of the baseline case of figure 7.4 has been reached with the single grid and the multigrid methods. The convergence rate has been improved by a factor of 8 by using the multigrid method. The convergence breakdown has not been produced, since the block-Jacobi preconditioning has been used.

Figure 7.8 shows that with the use of the block-Jacobi preconditioning the convergence rate

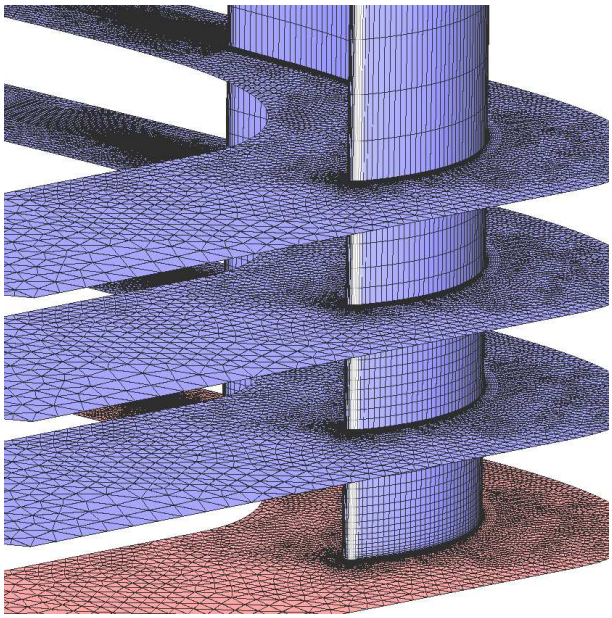


Figure 7.9: View of the three-dimensional semi-unstructured pattern of the discretisation of the T106 airfoil.

is independent of the number of grid points, as it was the case in the Euler case (see figure 7.2). Again, this means that for each grid level the high frequency error modes have been properly damped. It must be highlighted that for highly stretched cells, the block-Jacobi alone is not capable of damping all the high frequency error modes. For a correct behaviour of the whole algorithm, the multigrid coarser meshes should be constructed semi-coarsening the highly stretched cells, i.e., just agglomerating in the direction normal to the wall. That strategy is convenient to leave the high frequency error modes in the streamwise direction out of the multigrid strategy. These errors, that are not properly damped, should not be transmitted to the coarser grids to avoid its aliasing in low frequency errors that stall the convergence of the method. Without coupling these two strategies, block-Jacobi preconditioning and semi-coarsening, the multigrid algorithm fails to improve the convergence rate in cases with highly stretched cells.

By extruding the grid of figure 7.4 we obtain the three-dimensional grid of the T106 airfoil, that consists in 83 planes, that are highly clustered next to the end wall to capture the boundary layer gradients in that zone (see figure 7.9). The resulting grid has 996203 nodes. The multigrid agglomeration algorithm makes use of the extrusion pattern, and performs a two-dimensional agglomeration just in the hub, and then repeats the agglomerated grid in every grid plane, performing a one-dimensional agglomeration, as it has been explained in section 4.5.

The convergence to the steady state of this case is presented in figure 7.10 for scalar artificial viscosity, matricial artificial viscosity, and matricial artificial viscosity in conjunction with

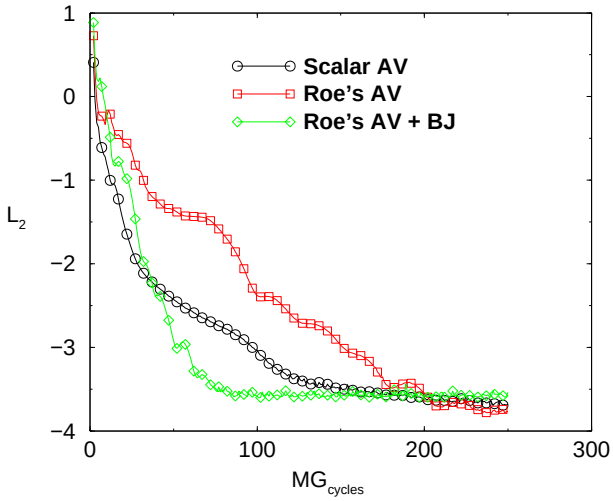


Figure 7.10: Influence of the artificial viscosity model in the convergence rate of a three-dimensional RANS simulation of the T106 airfoil using the $k-\omega$ turbulence model. 996203 nodes, multigrid V-cycle with 3 grid levels.

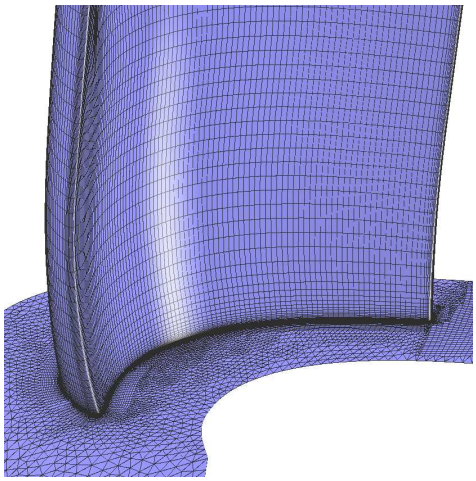


Figure 7.11: View of the Low Pressure Turbine Vane geometry.

block-Jacobi preconditioning. This result is similar to that depicted in figure 7.6, except for the fact that the convergence rate breakdown for the Roe's artificial viscosity without block-Jacobi preconditioning appears earlier. However, the main result, which is the benefit of using the matricial artificial viscosity in conjunction with the block-Jacobi preconditioning, is also recovered.

7.2. Low Pressure Turbine Vane

Figure 7.12 shows the effect of the block-Jacobi preconditioning on the convergence history of the low pressure turbine vane depicted in figure 7.11. This case has an aspect ratio $\Lambda \simeq 5$, an exit Mach number $M_{exit} \simeq 0.6$ and a Reynolds number based on the axial chord $Re_{Cx} \simeq 1.2 \times 10^5$. The domain has been meshed using a semi-unstructured mesh of 1249060 points with 95 mesh

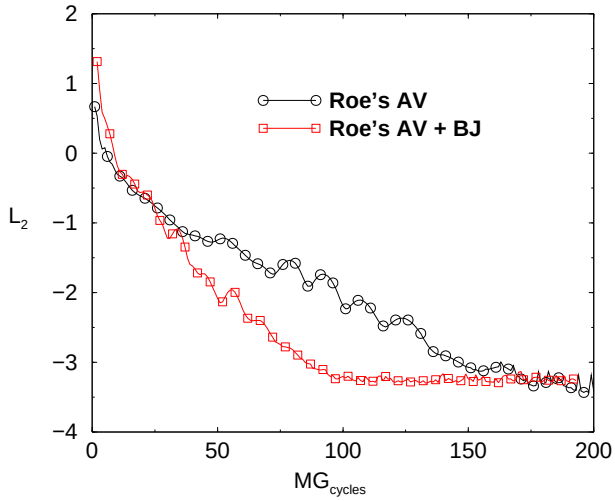


Figure 7.12: Influence of the artificial viscosity model in the convergence rate of a three-dimensional RANS simulation of a LPT vane using the $k-\omega$ turbulence model. 1249060 nodes. Multigrid V-cycle with 3 grid levels.

M_{exit}	0.68
Re_{Cx}	1.5×10^5
$\Delta\alpha_{torsion}$	0.5°
k	0.45

Table 7.2: Flow conditions for the ADTURB vibrating rotor blade.

planes and has been run in three processors. The behaviour is identical to that described in the previous sections, i.e., a better convergence rate when the block-Jacobi is enabled. The block-Jacobi case has taken 12 hours to reach the steady solution, using three Pentium IV at 3GHz.

7.3. Unsteady Results

The results of this section are based on the simulation of the ADTURB rotor blade [80], whose conditions are specified in table 7.2. The simulations are performed by combining the dual time step method described in chapter 6 with the steady state acceleration techniques expounded in chapters 3 and 5.

Table 7.3 shows the results for two different viscous grids of the rotor. The coarsest of them has been displayed in figure 7.13. The rotor vibrates in torsion mode with an amplitude $\Delta\alpha = 0.5°$

Grid points	No-DTS it./period	DTS equivalent it./period	Ratio
7089	17040	3900	4.37
67891	48612	7800	6.23

Table 7.3: Influence of the dual time step method in the number of iterations required to simulate one vibration period of the ADTURB rotor blade.

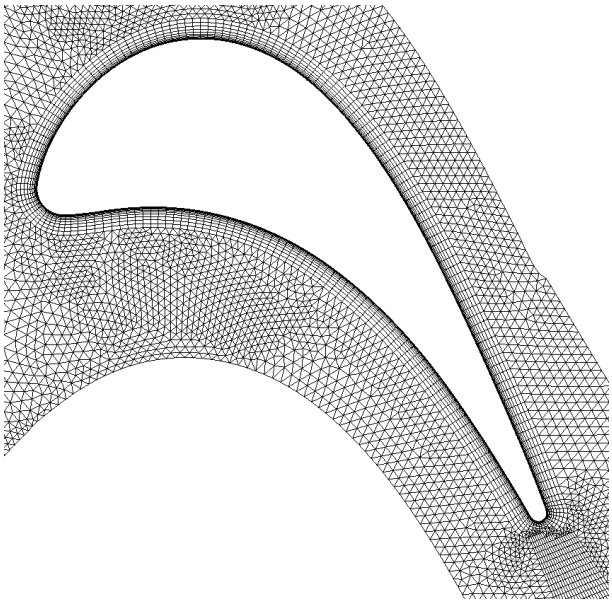


Figure 7.13: View of the grid used to discretise the ADTURB rotor blade. 7089 points.

and a reduced frequency 0.45. The benefits of the scheme in terms of number of iterations may be seen in table 7.3. The ratios are close to those obtained trying to converge the steady case with or without multigrid. That means that the same dual time step simulations performed without the preconditioned multigrid method would require a similar number of iterations than those performed with the explicit time marching scheme. Therefore, little improvement would be obtained when using the dual time step scheme without the steady acceleration capabilities. With this table we can conclude that dual time step best performance is produced for finer grids, where the multigrid method really increases the steady convergence rate. Obviously, these ratios are reduced frequency dependent, meaning that dividing by two the frequency doubles the number of required iterations per period when running without dual time step, but preserves the dual time step equivalent iterations, hence doubling the ratios presented in table 7.3. Thus for high frequencies the method is less effective.

Another way of increasing the ratios of table 7.3 is diminishing the number of iterations per time step. Full convergence of each time step is desired, and the number of iterations has to be large enough to ensure that, but we can reduce the convergence level occurs and observe the impact in the solution quality.

Results for this vibrating geometry are presented in fig.7.14, all of them for a 67891 point viscous grid. The computation of inter-blade phase angle $\sigma = 0^\circ$ required about 20 hours of CPU time on a 2 GHz P-IV, computing 5 periods, 12 time steps per period, 20 dual time step multigrid cycles per time step. On the other hand, the other two cases, that required the use of

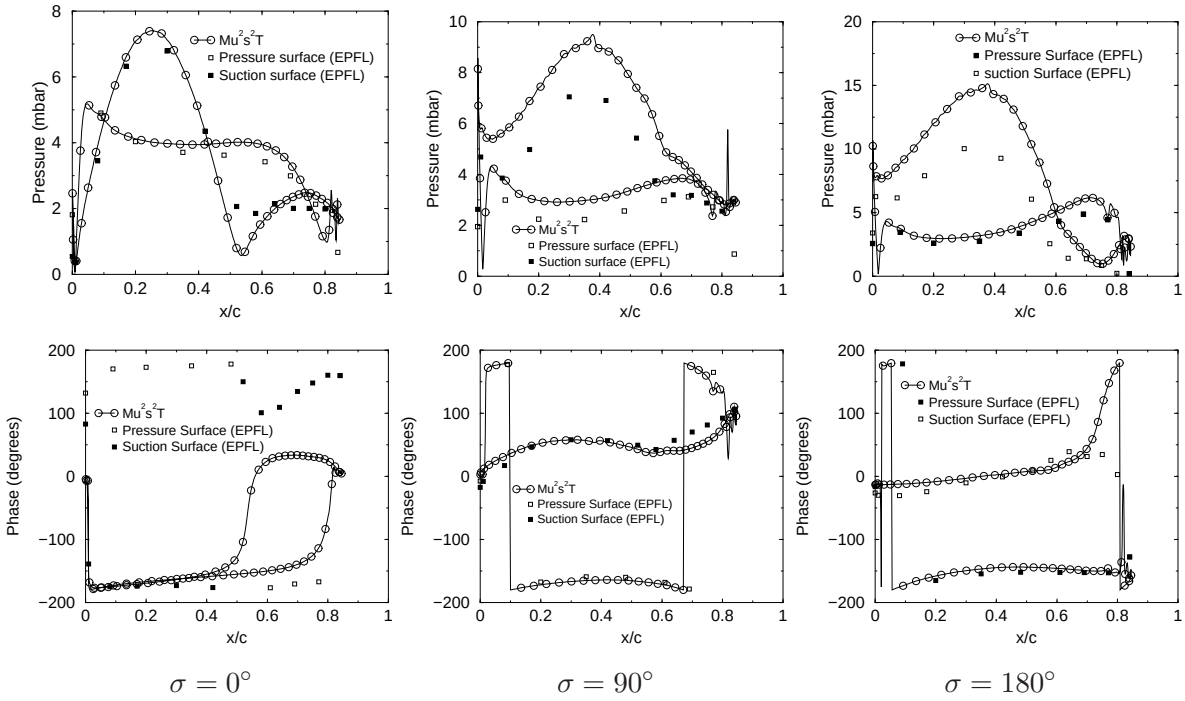


Figure 7.14: Unsteady Pressure Amplitude (top) and Phase (bottom) associated to the torsion vibration of the ADTURB rotor blade for three inter-blade phase angles. Fine Viscous Grid. 67891 points.

phase-lagged boundary conditions, needed more periods to reach a periodic movement. Results were obtained with 15 periods, therefore requiring about 60 CPU hours with the same 2 GHz P-IV processor. These simulations require about two weeks of computing time without the use of the dual time step method and therefore the savings associated to the dual time step are very large. The comparison with experimental results [80] is in general acceptable except for the phase of the $\sigma = 0°$ case.

Figure 7.15 displays the influence of the number of multigrid cycles per physical time step in the solution quality. All three simulations are performed with the grid of figure 7.13. With $n = 10$ iterations every time step is fully converged. Fifteen periods are simulated to reach the periodic solution, taking four hours in a 1.8 GHz P-IV. With $n = 5$ the error for each time step is an order of magnitude higher than for the with $n = 10$ iterations, but the solution is hardly modified. The computing time is reduced to 2 hours in the same computer. With $n = 3$ the error is about two orders of magnitude higher than for the fully converged case. The solution is slightly affected, mostly next to the trailing edge and also in the suction side pressure peak. But the overall result is nearly the same and about another factor of two in CPU time is saved (72 minutes are needed to run this simulation).

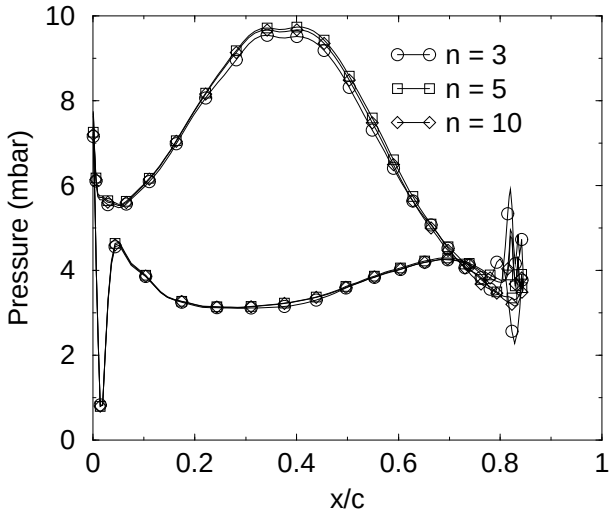


Figure 7.15: Influence of the number of multigrid cycles per Physical Time Step in the solution quality. 7089 points.

8. Adjoint Navier-Stokes equations

We have presented in the first part of the thesis the multigrid and the block-Jacobi preconditioning methods. Both techniques are used to accelerate the convergence of the Reynolds-averaged Navier-Stokes equations to the steady state. Within this context, it is realistic to tackle methodologies that refine a given design optimising an objective function at a reasonable computational cost.

The use of evolutionary or genetic algorithms for optimising turbomachinery blades is still prohibitive in terms of CPU time, requiring hundreds, if not thousands, of evaluations of the cost function to reach an optimum [31, 9]. Nevertheless, these algorithms provide valuable information, especially when doing multi-objective optimisation, and may become the only way to reach a global optimum when the objective function is noisy. However, for regular enough cost functions, gradient-based optimisation methods usually obtain the optimum with much less computational effort.

A gradient-based aerodynamic optimisation method requires two main building blocks: a non-linear Navier-Stokes solver to evaluate the proposed geometry and an efficient tool to compute the gradient. Efficient gradient evaluations of aerodynamic configurations are performed with the aid of adjoint Navier-Stokes equations solvers. Its use was pioneered by Pironneau [76] and later introduced by Jameson [42, 47, 50, 49], who used them first for 2D airfoil optimisations, then for wings and finally for complete aircraft configurations. Nowadays adjoint codes are widely used in optimisation problems in both the discrete and continuous approaches. In the continuous approach the adjoint equations are first formulated and then discretised in the computational domain [4]. The discrete approach, instead, linearises the discrete Navier-Stokes equations and develops the adjoint discrete problem from them [3, 28, 35]. This second approach is more suitable when a linear code exists [36], since the adjoint code routines can always be cross checked against their linear counterparts. The existence of a linearised version of the code [20] has favoured this latter approach in our case.

This chapter describes how the adjoint solver is derived from the baseline Navier-Stokes solver.

It will be shown in detail how the adjoint discretisation is derived, the physical meaning of the adjoint equations, and how they are solved.

8.1. Derivation of the Adjoint Discrete Equations

The optimisation problem consists in minimising a cost function $f(\mathbf{U}_j, \boldsymbol{\varphi}_k)$, where the variables $\mathbf{U}_j$ must fulfill the steady state discrete Navier-Stokes equations, that can be schematically written as

$$\mathbf{R}_i(\mathbf{U}_j, \boldsymbol{\varphi}_k) = 0, \tag{8.1}$$

and $\boldsymbol{\varphi}_k$ represents geometric parameters that are modified to control the objective function f. The restrictions imposed by the discrete Navier-Stokes equations can be absorbed by the functional, by multiplying each of them by a Lagrange multiplier, $\boldsymbol{\psi}_i$,

$$g(\mathbf{U}_j, \boldsymbol{\varphi}_k) = f(\mathbf{U}_j, \boldsymbol{\varphi}_k) + \boldsymbol{\psi}_i^T \cdot \mathbf{R}_i(\mathbf{U}_j, \boldsymbol{\varphi}_k) \tag{8.2}$$

As long as the steady state is fulfilled, the problems of minimising f and g are equivalent. The gradient of g is obtained by differentiating Eq. (8.2):

$$\frac{dg}{d\boldsymbol{\varphi}_k} = \left(\frac{\partial f}{\partial \mathbf{U}_j}\right)^T \frac{\partial \mathbf{U}_j}{\partial \boldsymbol{\varphi}_k} + \frac{\partial f}{\partial \boldsymbol{\varphi}_k} + \boldsymbol{\psi}_i^T \cdot \left(\left[\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_j}\right] \frac{\partial \mathbf{U}_j}{\partial \boldsymbol{\varphi}_k} + \frac{\partial \mathbf{R}_i}{\partial \boldsymbol{\varphi}_k}\right) \tag{8.3}$$

which shows us two strategies to proceed when evaluating the gradient:

1. The standard or classical approach is to linearise Eq. (8.1) to obtain

$$\left[\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_j}\right] \frac{\partial \mathbf{U}_j}{\partial \boldsymbol{\varphi}_k} + \frac{\partial \mathbf{R}_i}{\partial \boldsymbol{\varphi}_k} = 0 \tag{8.4}$$

   and remove in Eq. (8.3) the dependence of the Lagrange multipliers. The gradient of the modified objective function, g, is then

$$\frac{dg}{d\boldsymbol{\varphi}_k} = \left(\frac{\partial f}{\partial \mathbf{U}_j}\right)^T \frac{\partial \mathbf{U}_j}{\partial \boldsymbol{\varphi}_k} + \frac{\partial f}{\partial \boldsymbol{\varphi}_k} \tag{8.5}$$

   which states that the linear discrete Navier-Stokes equations must be evaluated for nominal variations of every geometric parameter, $\boldsymbol{\varphi}_k$, to obtain the flow sensitivities, $\partial \mathbf{U}_j / \partial \boldsymbol{\varphi}_k$. This is especially unsuitable for complex geometries where the number of geometric parameters required to define them is very large, of the order of hundreds.

2. Alternatively, the Lagrange multipliers can be chosen to remove the dependence on $\partial \mathbf{U}_j / \partial \boldsymbol{\varphi}_k$, giving rise to the need to solve the adjoint discrete Navier-Stokes equations

$$\left[\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_j}\right]^T \boldsymbol{\psi}_i + \frac{\partial f}{\partial \mathbf{U}_j} = 0 \tag{8.6}$$

to obtain the Lagrange multipliers. The gradient in Eq. (8.3) then yields

$$\frac{dg}{d\boldsymbol{\varphi}_k} = \boldsymbol{\psi}_i^T \cdot \frac{\partial \mathbf{R}_i}{\partial \boldsymbol{\varphi}_k} + \frac{\partial f}{\partial \boldsymbol{\varphi}_k} \tag{8.7}$$

which shows that one single evaluation of the adjoint equations can be used to determine the gradient by simply multiplying the adjoint variables, that is, the Lagrange multipliers $\boldsymbol{\psi}_i$, by the variation of the steady state equations with the geometric parameters, $\partial \mathbf{R}_i / \partial \boldsymbol{\varphi}_k$. This term may be evaluated using the complex variable method [36], that states that

$$\frac{\partial \mathbf{R}_i}{\partial \boldsymbol{\varphi}_k} = \lim_{\varepsilon \to 0} \frac{\Im \left[\mathbf{R}_i \left(\mathbf{U}_j, \boldsymbol{\varphi}_k + i\varepsilon\right)\right]}{\varepsilon} \tag{8.8}$$

and requires only one evaluation of the discrete Navier-Stokes equations, which consumes orders of magnitude less CPU time than the resolution of Eq. (8.4).

Equations (8.4) and (8.6) show that the linear and adjoint problems share the same eigenvalues, namely those of the matrix $[\partial \mathbf{R}_i / \partial \mathbf{U}_j]$, which is transposed in the adjoint problem. Therefore the asymptotic convergence of both problems must be the same when analogous iterative schemes are used.

8.2. Implementation of the Adjoint Discrete Equations

For the discrete approach, several tools that automatically generate the linear and adjoint codes from a programmed non-linear code [8, 24] are available. A review may be found in a report by Cusdin and Müller [25]. Even though automatic differentiation tools provide mathematically correct adjoint codes, their results are not optimal in terms of memory and CPU time, and they are often hard to trace and understand. Thus, it has been decided to hand-write the code following the recommendations provided by Giles [36, 34].

The process to write this code departs from the discrete RANS equations, written in Eq. (2.7). They are spatially discretised using an edge-based data structure, as presented in chapter 2. Then the terms of the discrete system of equations are linearised, yielding the linearised

semi-discrete RANS equations that are marched in time to obtain the steady state equations presented in equation (8.4), that can be written, making use of Eq. (2.7), as:

$$\left[\sum_{j=1}^{\#ed_i}\frac{1}{2}\left[\left(\frac{\partial \mathbf{F}_c}{\partial \mathbf{U}_k}-\frac{\partial \mathbf{F}_v}{\partial \mathbf{U}_k}\right)\Big|_i+\left(\frac{\partial \mathbf{F}_c}{\partial \mathbf{U}_k}-\frac{\partial \mathbf{F}_v}{\partial \mathbf{U}_k}\right)\Big|_j\right]\cdot \mathbf{n}_{ij}\sigma_{ij}\right]\frac{\partial \mathbf{U}_k}{\partial \boldsymbol{\varphi}_l} \quad + \\ +\frac{\partial}{\partial \boldsymbol{\varphi}_l}\left[\sum_{j=1}^{\#ed_i}\frac{1}{2}\left[(\mathbf{F}_c-\mathbf{F}_v)|_i+(\mathbf{F}_c-\mathbf{F}_v)|_j\right]\cdot \mathbf{n}_{ij}\sigma_{ij}\right] \quad = \quad 0 \tag{8.9}$$

In this equation, the contribution of the boundary edges has been omitted for the sake of clarity. The second term of the equation is evaluated making use of the complex variable method of equation (8.8). The first term can be computed analytically, since the dependence of the fluxes with the conservative variables $\mathbf{U}_k$ is known. Thus, the sum can also be expressed as a sum of known matrices

$$\left[\frac{\partial \mathbf{F}_{ij}}{\partial \mathbf{U}_k}\right]=\frac{1}{2}\left[\left(\frac{\partial \mathbf{F}_c}{\partial \mathbf{U}_k}-\frac{\partial \mathbf{F}_v}{\partial \mathbf{U}_k}\right)\Big|_i+\left(\frac{\partial \mathbf{F}_c}{\partial \mathbf{U}_k}-\frac{\partial \mathbf{F}_v}{\partial \mathbf{U}_k}\right)\Big|_j\right]\cdot \mathbf{n}_{ij}\sigma_{ij},$$

each one yielding the contribution of its corresponding edge to the linearised semi-discrete system of equations. Therefore, the matrix of the linear system of Eq. (8.4) can be expressed as:

$$\left[\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_k}\right]=\sum_{j=1}^{\#ed_i}\left[\frac{\partial \mathbf{F}_{ij}}{\partial \mathbf{U}_k}\right]$$

The equivalent matrix of the adjoint system of equations is obtained making use of Eq. (8.6), and noting that $[\sum A_i]^T=\sum A_i^T$. Thus, the matrix of the adjoint system of equations of Eq. (8.6) is:

$$\left[\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_k}\right]^T=\sum_{j=1}^{\#ed_i}\left[\frac{\partial \mathbf{F}_{ij}}{\partial \mathbf{U}_k}\right]^T$$

The next sections are devoted to obtain more concise expressions of the transposed matrices of the previous equation.

8.2.1. Adjoint Inviscid Fluxes

According to equation (8.9), the contribution of each edge to the value of the linearised inviscid fluxes is

$$\mathbf{LF}_{ij}^c=\left[\frac{1}{2}\left[\frac{\partial \mathbf{F}_c\cdot \mathbf{n}_{ij}}{\partial \mathbf{U}_k}\Big|_i+\frac{\partial \mathbf{F}_c\cdot \mathbf{n}_{ij}}{\partial \mathbf{U}_k}\Big|_j\right]\sigma_{ij}\right]\cdot\frac{\partial \mathbf{U}_k}{\partial \boldsymbol{\varphi}_l} \tag{8.10}$$

Then the contribution to the nodes i and j will be

$$\mathbf{LF}_i^c = \mathbf{LF}_{ij}^c; \qquad \mathbf{LF}_j^c = -\mathbf{LF}_{ij}^c \tag{8.11}$$

since $\mathbf{n}_{ij} = -\mathbf{n}_{ji}$. The flux contribution for the nodes i and j is written, making use of Eqs. (8.10) and (8.11):

$$\begin{bmatrix} \mathbf{LF}_i^c \\ \mathbf{LF}_j^c \end{bmatrix} = \frac{\sigma_{ij}}{2} \begin{bmatrix} \left.\dfrac{\partial \mathbf{F}_c \cdot \mathbf{n}_{ij}}{\partial \mathbf{U}_k}\right|_i & \left.\dfrac{\partial \mathbf{F}_c \cdot \mathbf{n}_{ij}}{\partial \mathbf{U}_k}\right|_j \\ -\left.\dfrac{\partial \mathbf{F}_c \cdot \mathbf{n}_{ij}}{\partial \mathbf{U}_k}\right|_i & -\left.\dfrac{\partial \mathbf{F}_c \cdot \mathbf{n}_{ij}}{\partial \mathbf{U}_k}\right|_j \end{bmatrix} \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_j \end{bmatrix} \tag{8.12}$$

where only the non-null entries of the vectors and matrices of the whole system of equations are presented. This convention will be maintained from now on. In this equation, $\mathbf{u}_i = \partial \mathbf{U}_i / \partial \boldsymbol{\varphi}_l$.

The adjoint inviscid flux operator is the transposed of the linear operator in Eq. (8.12):

$$\begin{bmatrix} \mathbf{AF}_i^c \\ \mathbf{AF}_j^c \end{bmatrix} = \frac{\sigma_{ij}}{2} \begin{bmatrix} \left.\dfrac{\partial \mathbf{F}_c \cdot \mathbf{n}_{ij}}{\partial \mathbf{U}_k}\right|_i^T & -\left.\dfrac{\partial \mathbf{F}_c \cdot \mathbf{n}_{ij}}{\partial \mathbf{U}_k}\right|_i^T \\ \left.\dfrac{\partial \mathbf{F}_c \cdot \mathbf{n}_{ij}}{\partial \mathbf{U}_k}\right|_j^T & -\left.\dfrac{\partial \mathbf{F}_c \cdot \mathbf{n}_{ij}}{\partial \mathbf{U}_k}\right|_j^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_i \\ \boldsymbol{\psi}_j \end{bmatrix} \tag{8.13}$$

The transposition of the flux matrix produces a change in the physics of the problem, since the system matrix is now the transposed of the original one. Nevertheless, the characteristic waves remain the same in the linear and adjoint problems, as pointed out above. A change in the sign of the derivative is also produced, therefore the waves of the adjoint problem and these of the linear problem travel in opposite senses. The expression of the transposed inviscid flux matrix of Eq. (8.13) is obtained in section B.1.

8.2.2. Adjoint viscous fluxes

When using an edge-based data structure, two edge loops are required to evaluate the viscous fluxes, the first for the evaluation of the gradients of the variables, and the second for the computation of the fluxes themselves. If we write the contribution of each edge to the gradient of the linearised variables, making use of Eq. (2.13), we obtain:

$$\mathbf{G}_{ij}^{(\xi)} = \frac{1}{2} \left(\mathbf{u}_i + \mathbf{u}_j \right) \sigma_{ij}^{(\xi)} n_{ij}^{(\xi)}$$

where $\xi = x, y, z$. By applying a flux splitting analogous to that of Eq. (8.11), we obtain the contribution of the edge to the gradient of the nodes i and j:

$$\begin{bmatrix} \mathbf{G}_i \\ \mathbf{G}_j \end{bmatrix}^{(\xi)} = \frac{1}{2} \begin{bmatrix} \vartheta_i \cdot I & 0 \\ 0 & \vartheta_j \cdot I \end{bmatrix}^{-1} \begin{bmatrix} I & I \\ -I & -I \end{bmatrix} \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_j \end{bmatrix} \sigma_{ij}^{(\xi)} n_{ij}^{(\xi)} \tag{8.14}$$

being I a 5×5 identity matrix. In the previous equation, the edge contribution has already been divided by the volume to present the whole gradient formulation in the same formula, even though that division is performed just once at the end of the edge loop, hence saving some operations. The adjoint gradient edge contribution is obtained by transposing the matrices of Eq. (8.14):

$$\begin{bmatrix} \mathbf{AG}_i \\ \mathbf{AG}_j \end{bmatrix}^{(\xi)} = \frac{1}{2} \begin{bmatrix} I & -I \\ I & -I \end{bmatrix} \begin{bmatrix} \vartheta_i \cdot I & 0 \\ 0 & \vartheta_j \cdot I \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\psi}_i \\ \boldsymbol{\psi}_j \end{bmatrix} \sigma_{ij}^{(\xi)} n_{ij}^{(\xi)},$$

since the transposition of a product of matrices $[A \cdot B]^T = B^T \cdot A^T$. This transposition also produces a change in the sign of the derivative, like that mentioned when computing the adjoint inviscid fluxes.

The second loop over edges yields the linearised viscous fluxes. The contribution of each edge ij to the matrix used to compute these fluxes is given by equation (8.9):

$$\left.\frac{\partial \mathbf{F}_v \cdot \mathbf{n}_{ij}}{\partial \mathbf{U}_k}\right|_{ij} = \frac{1}{2} \left[\left.\frac{\partial \mathbf{F}_v \cdot \mathbf{n}_{ij}}{\partial \mathbf{U}_k}\right|_{i} + \left.\frac{\partial \mathbf{F}_v \cdot \mathbf{n}_{ij}}{\partial \mathbf{U}_k}\right|_{j} \right] \sigma_{ij}$$

Therefore the linearised viscous fluxes can be expressed as:

$$\mathbf{LF}_{ij}^{v} = \sum_{\xi=x,y,z} B_{ij}^{(\xi)} \sigma_{ij} \cdot \nabla \mathbf{u}|_{ij}^{(\xi)} + \frac{1}{2} \Phi_{ij} \cdot (\mathbf{u}_i + \mathbf{u}_j)\, \sigma_{ij} \tag{8.15}$$

where

$$\Phi_{ij} = \sum_{\xi=x,y,z} \frac{\partial}{\partial \mathbf{U}_k} \left(B_{ij}^{(\xi)} \cdot \nabla \mathbf{U}|_{ij}^{(\xi)} \right)$$

The matrices $B_{ij}^{(\xi)}$ are derived from the viscous terms vector $\mathbf{F}_v \cdot \mathbf{n}_{ij}$ and contain terms that are affected by derivatives in the ξ direction (see appendix B for a detailed description of matrices $B_{ij}^{(\xi)}$ and Φ_{ij}). $\nabla \mathbf{u}|_{ij}$ and $\nabla \mathbf{U}|_{ij}$ are given by Eq. (2.15). The contribution of this edge to the

linear viscous fluxes is

$$\begin{bmatrix} \mathbf{LF}_i^v \\ \mathbf{LF}_j^v \end{bmatrix} = \frac{\sigma_{ij}}{2} \sum_{\xi=x,y,z} \begin{bmatrix} B_{ij}^{(\xi)} & B_{ij}^{(\xi)} \\ -B_{ij}^{(\xi)} & -B_{ij}^{(\xi)} \end{bmatrix} \Lambda^{(\xi)} \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_j \end{bmatrix} + \frac{\sigma_{ij}}{2} \begin{bmatrix} \Phi_{ij} & \Phi_{ij} \\ -\Phi_{ij} & -\Phi_{ij} \end{bmatrix} \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_j \end{bmatrix} \tag{8.16}$$

where

$$\Lambda^{(\xi)} = \sum_{mn=1}^{Nedges} \frac{\sigma_{mn}^{(\xi)} n_{mn}^{(\xi)}}{2} \begin{bmatrix} \vartheta_m \cdot I & 0 \\ 0 & \vartheta_n \cdot I \end{bmatrix}^{-1} \begin{bmatrix} I & I \\ -I & -I \end{bmatrix}$$

Actually, this equation has the implicit assumption that

$$\nabla \mathbf{u}|_{ij} = \frac{1}{2} \left(\nabla \mathbf{u}|_i + \nabla \mathbf{u}|_j \right)$$

Equation (2.15) shows that the edge gradient has a more complex expression, but it has not been used here to simplify the formulation. The complete formulation is found in appendix B. The transposed of Eq. (8.16) yields the contribution of the edge ij to the adjoint viscous fluxes:

$$\begin{bmatrix} \mathbf{AF}_i^v \\ \mathbf{AF}_j^v \end{bmatrix} = \sum_{\xi=x,y,z} \frac{\sigma_{ij}}{2} \left(\Lambda^{(\xi)} \right)^T \begin{bmatrix} \left(B_{ij}^{(\xi)} \right)^T & -\left(B_{ij}^{(\xi)} \right)^T \\ \left(B_{ij}^{(\xi)} \right)^T & -\left(B_{ij}^{(\xi)} \right)^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_i \\ \boldsymbol{\psi}_j \end{bmatrix} + \frac{\sigma_{ij}}{2} \begin{bmatrix} \Phi_{ij}^T & -\Phi_{ij}^T \\ \Phi_{ij}^T & -\Phi_{ij}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_i \\ \boldsymbol{\psi}_j \end{bmatrix} \tag{8.17}$$

where

$$\left(\Lambda^{(\xi)} \right)^T = \sum_{mn=1}^{Nedges} \frac{\sigma_{mn}^{(\xi)} n_{mn}^{(\xi)}}{2} \begin{bmatrix} I & -I \\ I & -I \end{bmatrix} \begin{bmatrix} \vartheta_m \cdot I & 0 \\ 0 & \vartheta_n \cdot I \end{bmatrix}^{-1}$$

This expression points out that the adjoint gradients are applied after the adjoint viscous matrix operator, i.e:

- We first perform an edge loop to compute two terms: an array of auxiliary variables $\mathbf{\Psi}^{(\xi)}$ ($\xi = x, y, z$) and a contribution to the adjoint viscous fluxes $\mathbf{AF}^{v,1}$.

$$\mathbf{\Psi}^{(\xi)} = \sum_{ij=1}^{Nedges} \frac{\sigma_{ij}}{2} \begin{bmatrix} \left(B_{ij}^{(\xi)} \right)^T & -\left(B_{ij}^{(\xi)} \right)^T \\ \left(B_{ij}^{(\xi)} \right)^T & -\left(B_{ij}^{(\xi)} \right)^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_i \\ \boldsymbol{\psi}_j \end{bmatrix} \tag{8.18}$$

$$\mathbf{AF}^{v,1} = \sum_{ij=1}^{Nedges} \frac{\sigma_{ij}}{2} \begin{bmatrix} \Phi_{ij}^T & -\Phi_{ij}^T \\ \Phi_{ij}^T & -\Phi_{ij}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_i \\ \boldsymbol{\psi}_j \end{bmatrix} \tag{8.19}$$

- Then, we compute the adjoint gradients of the $\mathbf{\Psi}^{(\xi)}$ variables, obtaining a second contribution to the adjoint viscous fluxes:

$$\mathbf{AF}^{v,2} = \sum_{mn=1}^{Nedges} \left[\sum_{\xi=x,y,z} \frac{\sigma_{mn}^{(\xi)} n_{mn}^{(\xi)}}{2} \begin{bmatrix} I & -I \\ I & -I \end{bmatrix} \begin{bmatrix} \vartheta_m \cdot I & 0 \\ 0 & \vartheta_n \cdot I \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{\Psi}_i^{(\xi)} \\ \mathbf{\Psi}_j^{(\xi)} \end{bmatrix} \right]$$

- Finally, the adjoint viscous fluxes are obtained:

$$\mathbf{AF}^v = \mathbf{AF}^{v,1} + \mathbf{AF}^{v,2}$$

The Eqs. (8.18) and (8.19) are detailed in section B.3 of the appendix.

8.2.3. Adjoint artificial viscosity fluxes

The artificial viscosity fluxes are also calculated by performing two edge loops, therefore the formulation will be similar to that expounded for the adjoint viscous fluxes. When evaluating the linearised artificial dissipation fluxes, the first edge loop is used to compute the pseudo-Laplacian of Eq. (2.11). The contribution of the edge ij to the pseudo-Laplacian of nodes i and j is:

$$\begin{bmatrix} \mathbf{pL}_i \\ \mathbf{pL}_j \end{bmatrix} = \begin{bmatrix} \#ed_i \cdot I & 0 \\ 0 & \#ed_j \cdot I \end{bmatrix}^{-1} \begin{bmatrix} -I & I \\ I & -I \end{bmatrix} \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_j \end{bmatrix} \tag{8.20}$$

The edge contribution to the adjoint pseudo-Laplacian is then:

$$\begin{bmatrix} \mathbf{ApL}_i \\ \mathbf{ApL}_j \end{bmatrix} = \begin{bmatrix} -I & I \\ I & -I \end{bmatrix} \begin{bmatrix} \#ed_i \cdot I & 0 \\ 0 & \#ed_j \cdot I \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\psi}_i \\ \boldsymbol{\psi}_j \end{bmatrix}$$

Since the pseudo-Laplacian is a symmetric operator, its transposed yields the same operator, therefore the adjoint pseudo-Laplacian and its linearised counterpart would be identical if the grid were uniform, i. e., if the number of edges that reach a node $\#ed_i$ were the same for all the nodes of the grid.

The contribution of the edge ij to the linearised artificial dissipation fluxes of nodes i and j

is computed making use of Eqs. (2.8) and (2.10):

$$\begin{bmatrix} \mathbf{LF}_i^{av} \\ \mathbf{LF}_j^{av} \end{bmatrix} = \begin{bmatrix} -|A_{ij}| & |A_{ij}| \\ |A_{ij}| & -|A_{ij}| \end{bmatrix} \left\{ S_{ij} \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_j \end{bmatrix} - \right.$$
$$\left. - \frac{1}{2}(1-\kappa)(1-S_{ij})\,\Upsilon \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_j \end{bmatrix} \right\}$$

where the pseudo-Laplacian has been substituted by the sum of its edge contributions, given in Eq. (8.20):

$$\Upsilon = \sum_{mn=1}^{Nedges} \begin{bmatrix} \#ed_m \cdot I & 0 \\ 0 & \#ed_n \cdot I \end{bmatrix}^{-1} \begin{bmatrix} -I & I \\ I & -I \end{bmatrix}$$

The contribution of the edge ij to the adjoint artificial viscosity fluxes is obtained by transposing the previous equation.

$$\begin{bmatrix} \mathbf{AF}_i^{av} \\ \mathbf{AF}_j^{av} \end{bmatrix} = S_{ij} \begin{bmatrix} -|A_{ij}|^T & |A_{ij}|^T \\ |A_{ij}|^T & -|A_{ij}|^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_i \\ \boldsymbol{\psi}_j \end{bmatrix} -$$
$$- \frac{1}{2}(1-\kappa)(1-S_{ij})\,\Upsilon^T \begin{bmatrix} -|A_{ij}|^T & |A_{ij}|^T \\ |A_{ij}|^T & -|A_{ij}|^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_i \\ \boldsymbol{\psi}_j \end{bmatrix}$$

where

$$\Upsilon^T = \sum_{mn=1}^{Nedges} \begin{bmatrix} -I & I \\ I & -I \end{bmatrix} \begin{bmatrix} \#ed_m \cdot I & 0 \\ 0 & \#ed_n \cdot I \end{bmatrix}^{-1}$$

Then the process to evaluate the adjoint artificial viscosity fluxes is:

- Evaluate the array auxiliary variables $\mathbf{\Phi}$ and the former contribution to the adjoint fluxes:

$$\mathbf{\Phi} = \sum_{ij=1}^{Nedges} \frac{1}{2}(1-\kappa)(1-S_{ij}) \begin{bmatrix} -|A_{ij}|^T & |A_{ij}|^T \\ |A_{ij}|^T & -|A_{ij}|^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_i \\ \boldsymbol{\psi}_j \end{bmatrix} \tag{8.21}$$
$$\mathbf{AF}^{av,1} = \sum_{ij=1}^{Nedges} S_{ij} \begin{bmatrix} -|A_{ij}|^T & |A_{ij}|^T \\ |A_{ij}|^T & -|A_{ij}|^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_i \\ \boldsymbol{\psi}_j \end{bmatrix}$$

  The value of the pressure-based swith S_{ij} (see Eq. (2.12)) is kept constant when solving the adjoint problem, i.e., it is not linearised.

- Compute the latter contribution to the adjoint artificial dissipation fluxes, by performing

the adjoint pseudo-Laplacian of the auxiliary variables $\mathbf{\Phi}$:

$$\mathbf{AF}^{av,2} = \sum_{mn=1}^{Nedges} \begin{bmatrix} -I & I \\ I & -I \end{bmatrix} \begin{bmatrix} \#ed_m \cdot I & 0 \\ 0 & \#ed_n \cdot I \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{\Phi}_i \\ \mathbf{\Phi}_j \end{bmatrix}$$

- The artificial dissipation fluxes are

$$\mathbf{AF}^{av} = \mathbf{AF}^{av,1} + \mathbf{AF}^{av,2}$$

The expression of Eq. (8.21) is obtained in section B.2 of the appendix.

8.3. Resolution of the Adjoint Discrete Equations

An appropriate iterative scheme for solving the discrete equations is needed. Since the linear and adjoint problem matrices share the same eigenvalues, a valid iterative scheme for the linear problem is also valid, in terms of stability, for the adjoint. Nevertheless, when a validated linear code exists, the use of an adjoint iterative scheme derived from the linear is helpful to predict the correctness of the implementation of the discrete adjoint equations. In each iteration, the linear and adjoint codes provide the same gradient value, ensuring that the asymptotic rate of convergence of both codes coincides. This approach has been followed, among others, by Giles et al. [34] and Nielsen et al. [71], and consists in finding the adjoint counterpart of the linear iterative scheme. If an iterative procedure with an associated matrix $[IT]$ for the linear problem (Eq. (8.4)) is used

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^n + [IT] \left(\left[\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_j} \right] \mathbf{u}_j^n + \frac{\partial \mathbf{R}_i}{\partial \boldsymbol{\varphi}_k} \right), \tag{8.22}$$

where $\mathbf{u}_j^0 = 0$ is the initial condition, it can be demonstrated [33] that the analogous adjoint iterative scheme is

$$\boldsymbol{\psi}_j^n = \boldsymbol{\psi}_j^{n+1} + [IT]^T \left(\left[\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_j} \right]^T \boldsymbol{\psi}_i^{n+1} + \frac{\partial f}{\partial \mathbf{U}_j} \right) \tag{8.23}$$

with $\boldsymbol{\psi}_i^N = 0$. Eqs. (8.22) and (8.23) show that the adjoint equation is integrated marching backwards in time, from iteration $n = N$ to $n = 0$, and that the matrix associated to the adjoint iterative scheme is the transposed of the linear one. This matrix $[IT]$ could represent any kind of iterative method, from a Runge-Kutta scheme to a multigrid method. The formulation of the Runge-Kutta multistage scheme with partial updates of the viscous terms, the residual

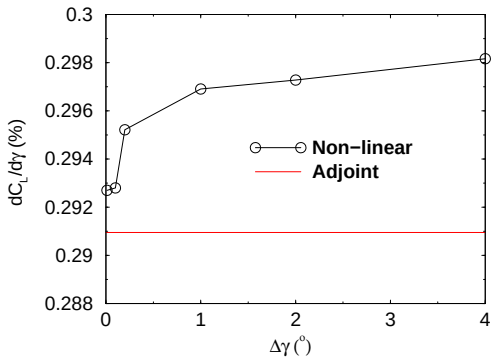


Figure 8.1: Sensitivity of the lift coefficient to the stagger angle. T106 blade ($M_{is,exit} = 0.59$).

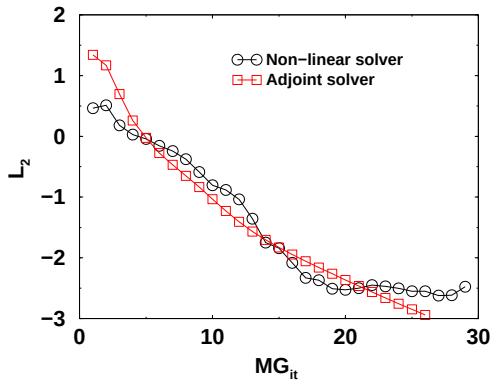


Figure 8.2: Comparison of the convergence history of the T106 case for the non-linear and adjoint solvers.

smoothing operator and the intergrid transfer operators for the multigrid method are developed in section B.5.

8.4. Code validation

The adjoint code implementation has been debugged with the aid of an in-house linear solver [20], by ensuring that $\boldsymbol{\psi}^T [A] \mathbf{u} = \mathbf{u}^T [A]^T \boldsymbol{\psi}$, where the matrix $[A]$ represents a generic operator, e.g.: the spatial discretisation, multigrid operators, etc. On the other hand, we have compared the sensitivities obtained by the adjoint code with those obtained by numerical differentiation, using the non-linear version of the code.

The sensitivity of the lift coefficient, C_L, to variations in the stagger angle, γ, has been assessed for a two-dimensional viscous solution of the T106 blade [87] ($\gamma = 59.28°$, $M_{is,exit} = 0.59$).

These results are shown in figure 8.1. The sensitivities obtained by the adjoint code and by a second order differentiation scheme are very similar for both approaches. The discrepancies could be attributed to the incomplete linearisation of the artificial viscosity terms or to the lack of linearisation of the turbulence model, since the turbulent viscosity is kept constant in the

adjoint code.

Figure 8.2 compares the convergence of the T106 case for the non-linear and adjoint solvers. The convergence rate of both codes is about the same. However, the asymptotic convergence rate is more clearly seen in the adjoint solver probably due to its linear nature. This result was expected, since the linear and the adjoint problems share the same eigenvalues and thus the asymptotic convergence rate should coincide. In this case, 20-30 multigrid iterations are enough to converge the problem for engineering purposes. The CPU cost to obtain the non-linear solution and the adjoint solution is roughly the same.

9. Optimisation of non-axisymmetric end walls

Once the building blocks of the optimisation method have been presented, we could choose a cost function and a set of control parameters to optimise it. One of the main targets of the three-dimensional design of vanes, and to a lesser extent, of rotor blades, is the minimisation of secondary losses, measured both in terms of total pressure losses and secondary kinetic energy. This goal can be achieved by modifying the axial and tangential lean of the airfoil and/or the turbine end walls. As a preliminary use of the adjoint Navier-Stokes solver as part of an optimisation algorithm, we have focused in the additional reduction in losses that may be obtained by removing the constraint that the end wall is axisymmetric. It has been shown that by altering the end wall shape it is possible to modify the secondary vortex system and reduce the vortex strength downstream of the blade rows [79, 40, 81]. This reduction impacts on both total pressure losses and secondary kinetic energy, but it has been observed that the sensitivity of the latter is larger. The end walls are perturbed with a series of sines and cosines at different axial locations of the row, as shown in figure 9.1, whose amplitudes need to be determined during the design phase of the airfoil.

The design of a profiled end wall involves tens of parameters and may become a huge task

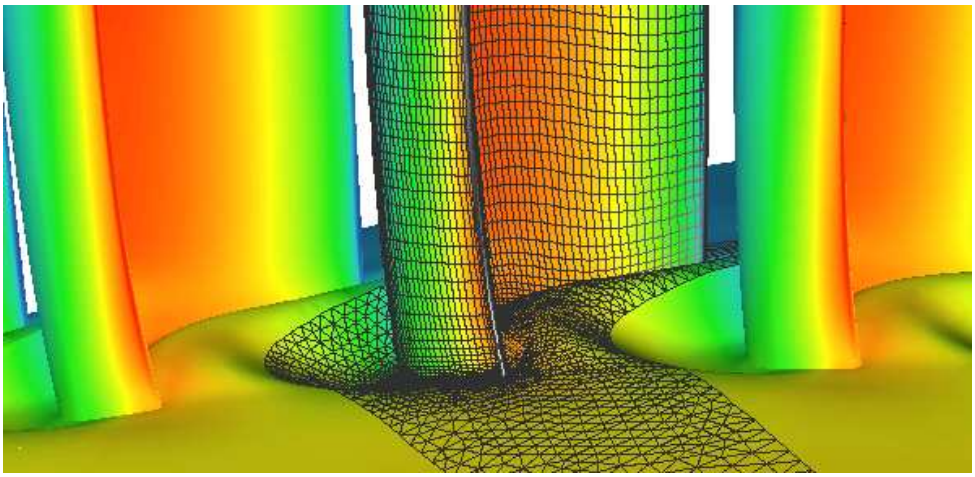
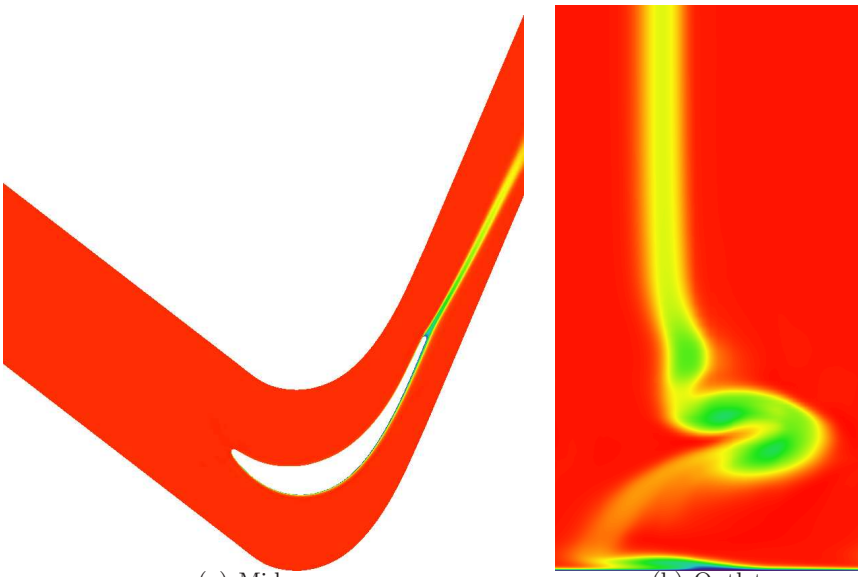


Figure 9.1: Detail of an end wall design.



(a) Midspan (b) Outlet

Figure 9.2: Colour plot of the stagnation pressure for a T106 airfoil simulation at different locations.

for the designer if it is not properly automatised. In its most primitive form the designer has to obtain the sensitivities of the secondary losses with regard to the design parameters, which means tens of evaluations of the cost function, each requiring the solution of the three-dimensional Reynolds-Averaged Navier-Stokes Equations in the flow domain, discretised with a grid of about 10^6 points. The designer has to choose a proper combination of the perturbations and evaluate again the design. The designer not only has to monitor the secondary losses but also traverse the solution to find local separations and other unexpected features. The automatisation of this task would not only speed-up the design but it could potentially improve the final solution.

9.1. Secondary flow pattern

This section briefly presents the secondary vortex system in a low pressure turbine. Secondary losses are generated due to the interaction between the end wall boundary layer and the pressure gradient generated by the airfoils. When the flow is two-dimensional, the losses are produced in the airfoil boundary layer (see figure 9.2a). Far away of the end wall boundary layer the fluid velocity is large and the transversal pressure gradient produced in the channel by the presence of the airfoils hardly modifies the streamlines, that essentially follow the airfoil walls. The situation is completely different when we consider the end wall boundary layer. Here, the fluid momentum is very low and the transversal pressure gradient provokes a flow migration from the pressure to

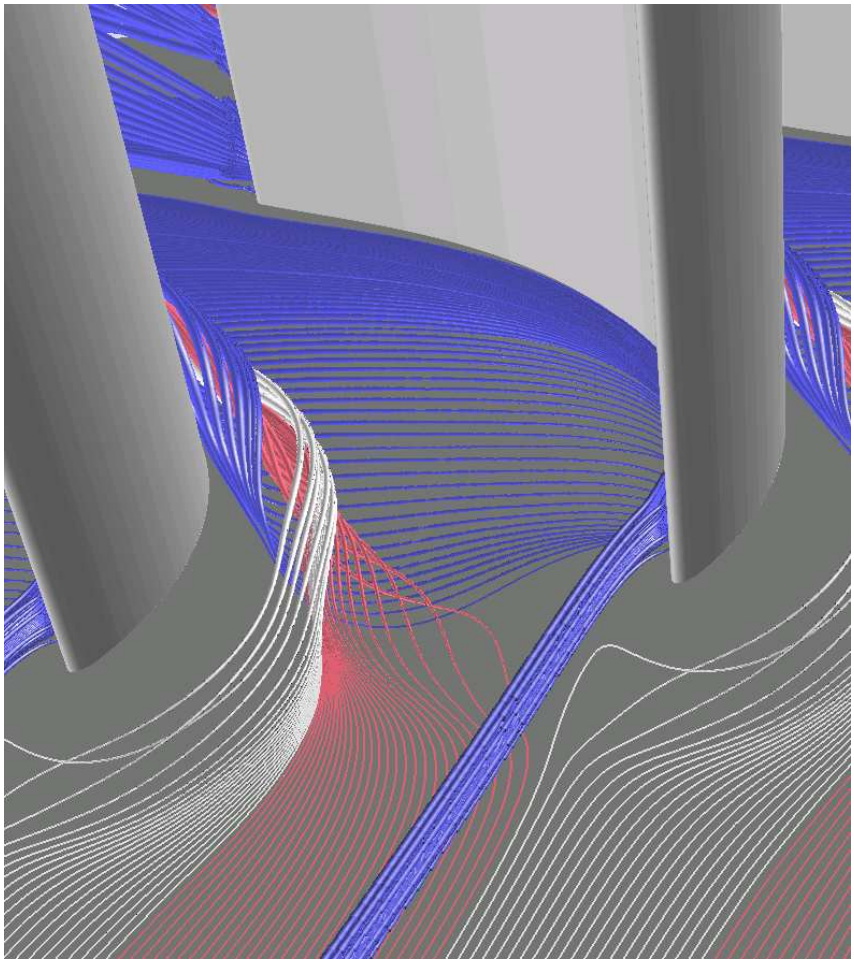


Figure 9.3: Detail of the end wall streamlines for a T106 airfoil simulation.

the suction side of the airfoil, giving rise to a series of vortices, that conform the typical structure of the secondary flows, depicted in figure 9.2b. We can distinguish two main vortices that yield the largest contribution to the secondary kinetic energy: the horseshoe vortex and the passage vortex. They are represented in figure (9.3) by means of streamlines that lie in the end wall region of a linear T106 airfoil. The white and indigo isolines represent the pressure side and the suction side legs of the so called horseshoe vortex. The vortex is produced when the pressure rise due to the presence of the airfoil forces the inlet boundary layer to split well ahead of the leading edge to circumvent the airfoil. The point where the legs of the horseshoe vortex separate is known as the saddle point. The pressure side leg of the horseshoe vortex moves towards the suction side of the airfoil due to the pressure gradient, yielding a clockwise rotating vortex, that moves up due to the adverse pressure gradient produced by the proximity of the airfoil wall. The other leg of the horseshoe vortex yields a counter-clockwise rotating vortex that is also raised when it approaches the airfoil. The streamlines of the passage vortex, which have been coloured in blue, come from outside the boundary layer. These streamlines move towards the end wall after impinging the leading edge just above the horseshoe vortex. Once there, the transversal pressure gradient pushes them towards the suction side of the airfoil, giving rise to a clockwise rotating vortex, referred to as the passage vortex.

9.2. Cost function definition

Keeping in mind that the final target is to minimise the production of secondary losses maintaining the mean flow at the row exit unaltered, a proper cost function needs to be chosen. The cost function selected to drive the optimisation is the mass averaged $SKEH$, that is the product of the secondary kinetic energy and the helicity, at an axial plane cut. The non-dimensional secondary kinetic energy for a node i is defined as

$$SKE_i = \frac{(\mathbf{v}_i - \mathbf{v}_{pi})^2}{\mathbf{v}_{exit}^2} \tag{9.1}$$

where $\mathbf{v}_{exit}$ is the exit mass averaged velocity, and $\mathbf{v}_{pi}$ is the projection of the velocity vector at the i node over the circumferentially mass averaged velocity, $\mathbf{v}_m$,

$$\mathbf{v}_{pi} = \frac{\mathbf{v}_i \cdot \mathbf{v}_m}{\mathbf{v}_m^2}\mathbf{v}_m$$

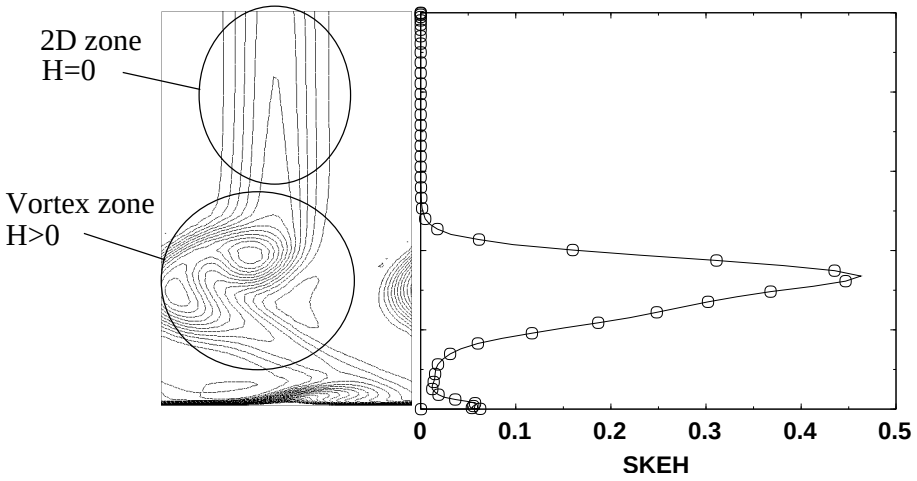


Figure 9.4: Secondary flow pattern for a 3D straight cascade. Left: Total pressure iso contours. Right: Mass averaged non-dimensional $SKEH$ distribution.

being $\mathbf{v}_i$ the local velocity. The non-dimensional helicity, H_i, is defined as

$$H_i = \frac{|\mathbf{v}_i \cdot \boldsymbol{\omega}_i|}{\mathbf{v}_{exit}^2/\ell_c} \tag{9.2}$$

where $\boldsymbol{\omega}_i$ is the local vorticity vector and ℓ_c is a characteristic length, typically the blade chord. The non-dimensional $SKEH$ value is obtained multiplying equations (9.1) and (9.2)

$$SKEH_i = \frac{(\mathbf{v}_i - \mathbf{v}_{pi})^2 \, |\mathbf{v}_i \cdot \boldsymbol{\omega}_i|}{\mathbf{v}_{exit}^4/\ell_c} \tag{9.3}$$

The aim of multiplying the SKE_i by the helicity is clearly seen in a linear cascade. In such a configuration the secondary flow is confined next to the end wall. At the cascade midspan, the wake velocity deficit gives rise to a vorticity vector which is perpendicular to the velocity (see figure 9.4), the helicity vanishes and the wake contribution to the $SKEH$ is null. In the region where the secondary vortex is located, $\mathbf{v}_i$ and $\boldsymbol{\omega}_i$ are nearly aligned, therefore $H > 0$, contributing to increase the $SKEH$. Therefore, multiplying the SKE by the helicity isolates the phenomenon we want to minimise, which is the strength of the vortex.

End wall perturbations may give rise to large adverse pressure gradients that may eventually promote massive separations of the flow. These detached bubbles are essentially a two-dimensional feature and hence the $SKEH$ is unable to detect them since $\mathbf{v}_i \cdot \boldsymbol{\omega}_i \simeq 0$. However, they induce large variations of the swirl angle. Even if the flow is attached, the swirl angle needs to be limited somehow, otherwise it will increase the losses in the downstream row, since changes in the swirl angle will give rise to changes in the incidence that will not be possible to absorb modifying the geometry of the downstream row. To account for these phenomena in the

optimisation process, we have introduced a penalty function that grows exponentially with the difference in the swirl angle between the baseline and the actual case. It is expressed as

$$F_p = f_1 \cdot e^{-\frac{\overline{\beta}_s - \beta_t + f_2}{f_3}}, \tag{9.4}$$

being f_1, f_2 and f_3 factors to conform the shape of the penalty function, β_t the target swirl angle and $\overline{\beta}_S$ the mass averaged swirl angle in the region of interest. Thus, when $\overline{\beta}_S < \beta_t$, the penalty function grows, and the new cost function $\overline{SKEH} + F_p$ moves further away from the minimum. This cost penalty function as well as the $SKEH$, and the derivatives of them, are obtained in appendix C.

9.3. Optimisation Method

The optimisation method used in this work consists of a projected gradient search combined with a Broyden's method to improve the final convergence to the desired optimum [54]. This method has been selected because its simplicity and capability to deal with complicated constraints that are not known a priori, i.e., constraints that require the complete simulation of the system to know if they are satisfied. This kind of constraints are frequent in turbomachinery problems, e.g., a fixed mass flow or a fixed power. Another important advantage of this method is that it requires only the computation of the functions involved and its gradients and no Hessian matrices are needed.

9.4. Designing LPT end walls

The adjoint code has been used to compute the derivatives of the $SKEH$ with regard to the geometry changes. The geometry of the end walls is modified by perturbing the axisymmetric baseline surface with a Fourier series:

$$P(x,\theta) = C(x) + \sum_{j=1}^{j=n_F} \left[A_j(x) \sin\left(j2\pi\frac{\theta}{\theta_b}\right) + B_j(x) \cos\left(j2\pi\frac{\theta}{\theta_b}\right) \right] \tag{9.5}$$

where θ_b is the blade pitch. The perturbations are specified at some fixed axial locations, which remain constant during the whole optimisation process. Six axial locations have been perturbed using the previous expression.

Two different optimised solutions are presented for the hub end wall of a LPT vane, one with

a single harmonic perturbation ($n_F = 1$), which is compared with a manual design with the same degrees of freedom, and another with multiple harmonics ($n_F = 4$). All the perturbation amplitudes are restricted to a maximum value of a 10% of the axial chord.

This case has an aspect ratio $\Lambda \simeq 5$, the exit Mach number is approximately 0.6 and the Reynolds number based on the exit conditions and the axial chord is 1.2×10^5. The domain has been discretised using a semi-unstructured mesh of 590.000 points. The construction of this mesh involves the radial smoothing of a two-dimensional grid previously constructed along specified radial planes [17]. Hence, obtaining the perturbed grids is a time consuming task that influences the overall optimisation time when the number of design parameters grows. In the next section we give some hints on the time spent on computing the cost function gradient comparing it with the overall optimisation time.

9.4.1. Computational cost.

All the cases presented in this chapter have been computed in two PIV @ 3GHz. The process is fully parallelised except for the construction of the grid. Both the manual design and the single harmonic optimised case consider the same number of design parameters, namely 18.

The manual design consists on the evaluation of the gradient by solving the Navier-Stokes equations as many times as the number of parameters, and then computing the gradient components by finite difference. The Navier-Stokes equations are evaluated twice for each parameter, with different values of the perturbation, to keep the non-linear sensitivity of the cost function to variations in the value of the design parameters. Once the gradient is calculated, the new geometry is evaluated, but just another optimisation step is performed, and only the most representative parameters (i.e., those with larger sensitivities) are varied. The whole design requires about 400 hours of CPU time to reach a solution (50 runs of the non-linear solver), but the whole process is not fully automatic and the designer has to drive the optimisation, increasing the design time even more. The optimiser consumes 60 hours of CPU time to reach the final solution (8 cost function evaluations), that supposes roughly an order of magnitude less time than the manual process. Besides, the process is fully automatised, avoiding the designers the tedious task of manually post-processing all solutions.

When considering the multiple harmonics case, the final solution is obtained in about 80 hours, which represents a 30% more CPU time, even though the number of parameters has been increased from 18 to 54. Since the number of optimiser iterations for both problems is

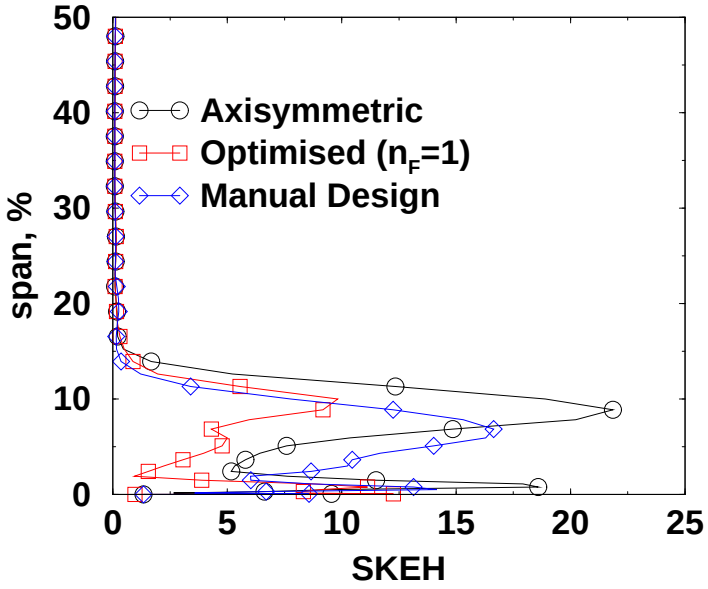


Figure 9.5: Comparison of the baseline, optimised solution and manual design cases. Mass averaged non-dimensional $SKEH$ at the outlet for the single harmonic case.

the same, the increase in the CPU time is produced due to the larger number of parameters. Hence, the construction of the semi-unstructured grid for each set of perturbation parameters noticeably influences the total CPU time, but its impact is smaller than the evaluation of 18 or 54 additional cost functions per gradient evaluation, which is another advantage derived from the use of the adjoint Navier-Stokes solver.

9.4.2. Single harmonic perturbation.

The solution obtained by the optimiser is compared with that proposed by the aerodynamic designer. The main difference between both approaches is that during the manual design, smaller maximum perturbation amplitudes have been allowed, and hence the flow is not as much disturbed as it is in the optimised solution. There are also flow effects that are taken into account when performing the manual design that can not be easily controlled by the optimiser, such as suction side flow separation close to the hub. This effect can only be indirectly addressed in the optimisation process by means of the swirl angle penalty function of Eq. (9.4).

The radial distribution of $SKEH$ for the axisymmetric baseline case and the profiled end walls may be seen in figure 9.5. A pure two-dimensional region with $SKEH \simeq 0$ between the 15% and 50% of the span is clearly distinguished. It may be seen that the $SKEH$ is appreciably reduced in the optimised solution, with the peak value divided approximately by a factor of two. The reduction is smaller for the manual design, due to the smaller amplitude of the perturbations, but the manual design allows a better control of the solution downstream of the perturbations.

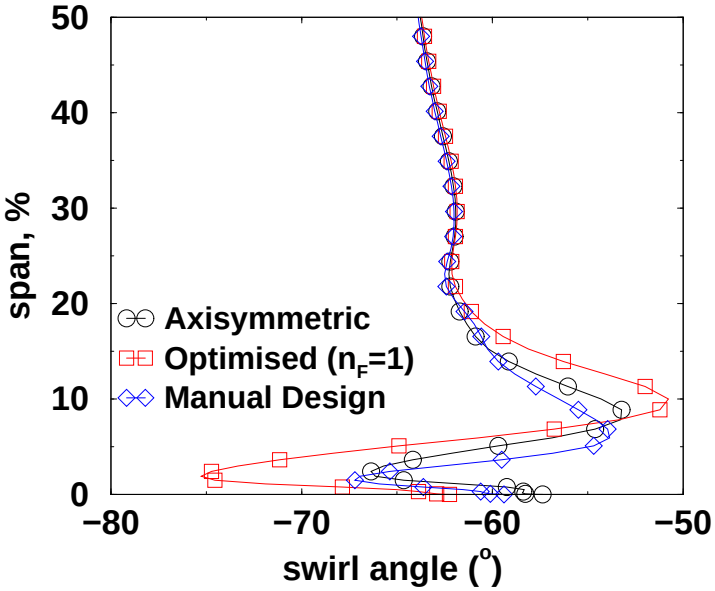


Figure 9.6: Comparison of the baseline, optimised solution and manual design cases. Mass averaged swirl angle at the outlet for the single harmonic case.

Both the swirl angle (figure 9.6) and the total pressure (figure 9.7) are improved in the manual design, while the same is not true for the optimised solution. Although the swirl angle is modified with respect to the base case, the variations are controlled by the penalty function. A more restrictive selection of the penalty function parameters would produce a swirl angle distribution closer to the baseline case. The total pressure losses have increased with the proposed design. This is an indication of the separation that takes place in the trailing edge region next to the hub and it cannot be directly controlled by the optimiser since there is no mechanism to reflect its impact on the penalty function. Thus, it is possible to weaken the vortex strength reducing the secondary losses, at the expense of increasing simultaneously the primary losses. This situation could be avoided by redefining the cost function. A reliable loss indicator that reflects both the total pressure loss and the mixing of the flow is the mixed-out average of the total pressure [77], but it has not been used in this work to maintain the same cost function used in manual designs.

Figure 9.8 represents the flow migration in the hub for this case, that is certainly unusual. This may be noticed by looking at the slope of the streamlines close to the hub, that head directly to the suction side. The differences in the migration patterns between the baseline and optimised cases are clear. The flow migration for the optimised case is much weaker than in the original, except when approaching the blade trailing edge. There, it gets stronger due to the almost complete disappearance of the suction side leg of the horseshoe vortex and the passage vortex, that in the axisymmetric case prevent the flow from turning too much in that region. This effect is also seen in the manual design, but it is weaker due to the smaller amplitude of the

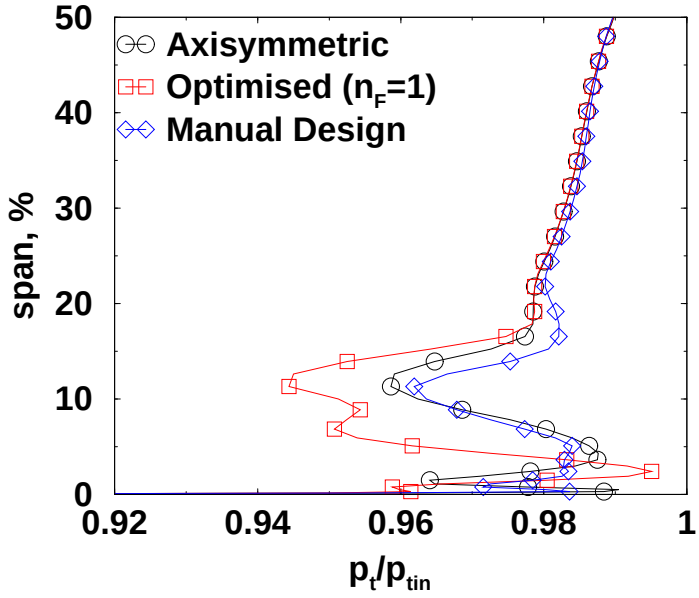


Figure 9.7: Comparison of the baseline, optimised solution and manual design cases. Mass averaged non-dimensional total pressure at the outlet for the single harmonic case.

end wall perturbations. In that design, the suction side vortex is still present. Also, the saddle point of the optimised case has moved towards the suction side, the blade is less front-loaded and the horseshoe vortex is weakened. This effect has not been reproduced in the manual design.

The pressure distribution in the blade-hub intersection for the optimised single harmonic case (figure 9.9 (left)) shows a reduction of the loading between the leading edge and A, due to the change of incidence, a pressure rise in the suction side at B, followed by a sharp decrease that ends at C, both associated to the bump located close to the suction side, then the flow separates (between C and D), and finally the pressure sharply rises again. The perturbations intended to be applied just for the suction side also modify the pressure side distribution, and the flow separates at the rear part of the blade due to the adverse pressure gradient. These adverse effects may be caused by a poor control of the end wall geometry due to an insufficient number of harmonics, and may be minimised with the use of a higher number of control parameters. The manual design roughly follows the same geometry obtained with the optimiser. There are two positive bumps, one next to the pressure side and the other above the suction side next to the trailing edge, and one negative bump, above the suction side at the beginning of the perturbation. The negative bump placed near the trailing edge in the pressure side is not obtained with the optimiser. Since the amplitudes of the perturbations for the manual design are smaller than for the optimised case, the effect of the non-axisymmetric end wall in the pressure distribution is less noticeable, as it occurs in the other distributions (figures 9.5 to 9.7).

9.4.3. Multiple harmonic perturbation.

The multiple harmonic perturbation case consists of four Fourier harmonics for each axial location, yielding a total number of 54 control parameters for the end wall. The axial locations of the perturbations have remained constant.

By adding more control parameters, all the distributions presented in the single harmonic case are improved. Figure 9.10 compares the final $SKEH$ distribution with the base solution. It is seen that the core distribution is greatly modified. The peak value has been divided by four and displaced towards the hub at 5% of the span. The swirl angle underturning has also been reduced by 2° in the core region of the span (see figure 9.11), while maintaining acceptable overturning values next to the hub, due to the use of the penalty function. The total pressure distribution (figure 9.12) shows the same trend as the $SKEH$. The peak of the losses has been displaced towards the hub about a 7% and, contrary to the previous case, its value has not been increased. The pressure distribution in the blade-hub intersection shows how it is possible to decouple the effects of the pressure and suction side of the blade by adding more control parameters in the circumferential direction. Thus, in Fig. 9.13 we see how the pressure side distribution is not affected very much except for the pressure bump around D. The suction side distribution shows an overall behaviour similar to the single harmonic case, but eliminating the undesirable bumps, that are seen in the latter. A pressure drop between A and B is followed by a rise at C, but the difference between the multiple and single harmonic distribution is that the sharp pressure rise next to the trailing edge (after D in Fig. 9.9) is not produced. Hence, the flow separation that leads to higher pressure losses in the single harmonic case is minimised.

The optimiser (figure 9.13, left) has placed four bumps across the channel, whose effect is the isolation of the vortices generated in the passage. This effect reduces the interaction of the vortices with the main stream and weakens their strength. The migration of the hub boundary layer (figure 9.14) also shows this phenomenon. We see how the streamlines are guided through the bumps in the rear part of the channel and how the suction side vortices are confined next to the suction side wall.

Both the single and the multiple harmonic solutions consider fixed axial locations for the geometry perturbations. The solutions could be enhanced if the number of axial stations were increased to improve the end wall control. By placing them in zones where the adjoint solution has larger values, such as next to the leading edge (see figure 9.15), smaller geometry changes could lead to larger changes in the flow configuration. However, this issue has not been addressed

in this work to keep the same restrictions as the designers.

Figure 9.8: Detail of the hub boundary layer flow migration for the single harmonic case. Top: axisymmetric. Middle: optimised solution. Bottom: manual design.

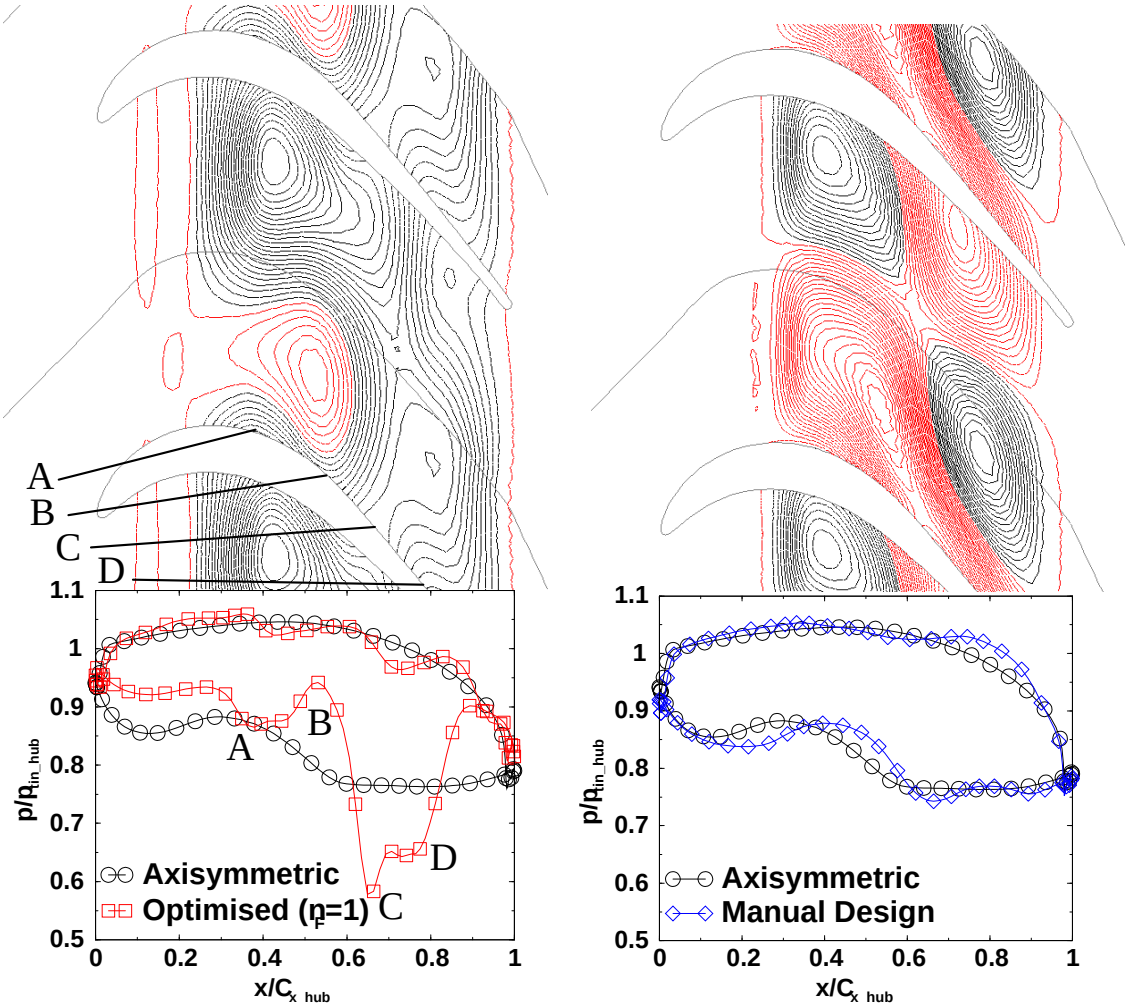


Figure 9.9: Isolines (black positive and red negative) of the hub surface perturbation (top) and comparison of the baseline and optimised non-dimensional pressure distribution on the blade-hub intersection (bottom) for the single harmonic case (left) and for the manual design (right).

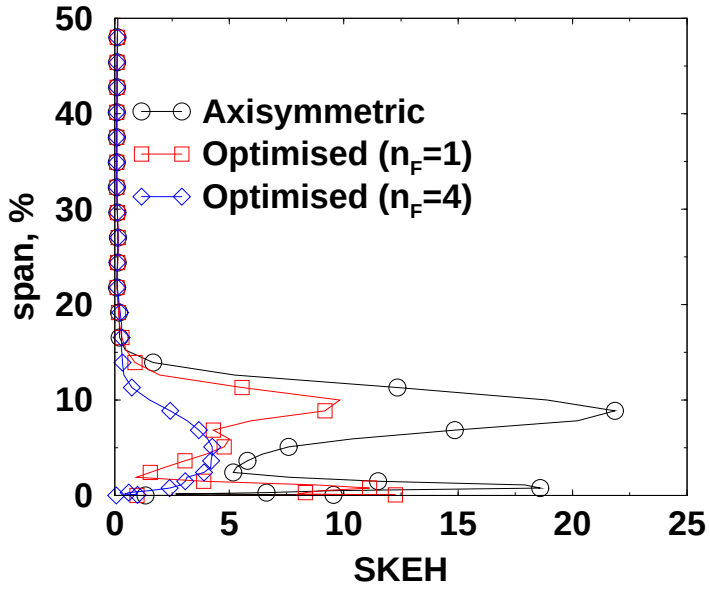


Figure 9.10: Comparison of the baseline and optimised mass averaged non-dimensional $SKEH$ at the outlet for the multiple harmonic case.

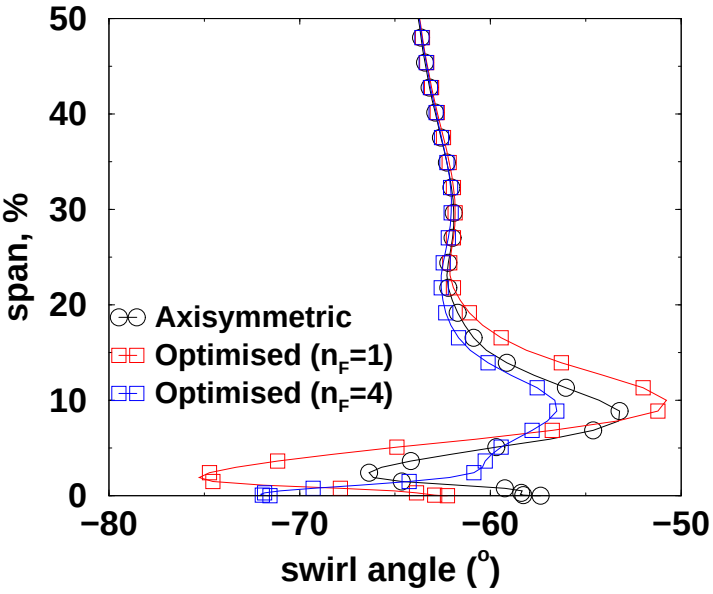


Figure 9.11: Comparison of the baseline and optimised mass averaged swirl angle at the outlet for the multiple harmonic case.

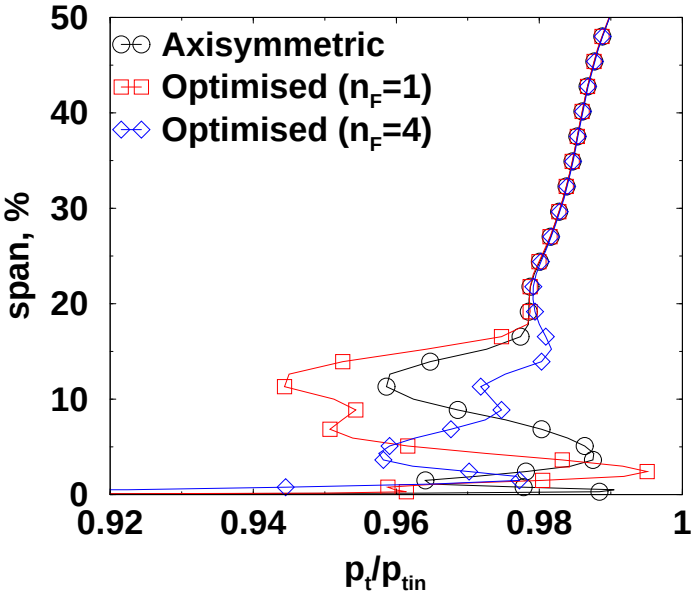


Figure 9.12: Comparison of the baseline and optimised mass averaged non-dimensional total pressure at the outlet for the multiple harmonic case.

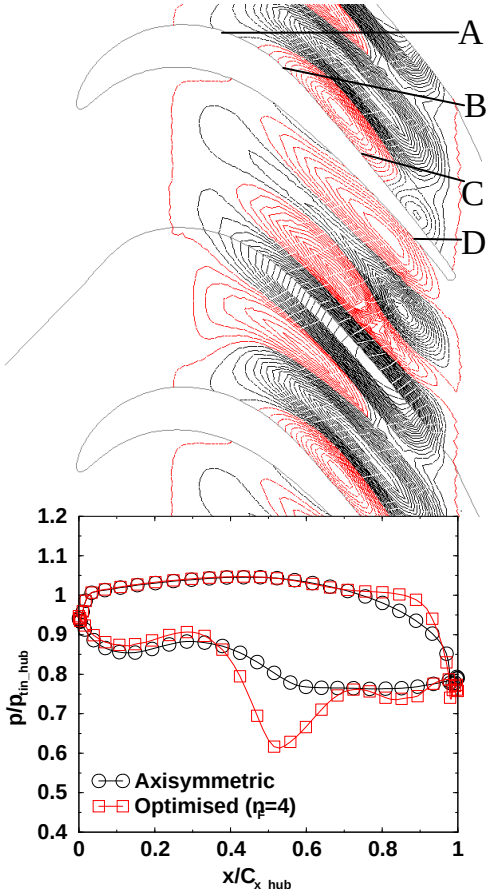


Figure 9.13: Isolines (black positive and red negative) of the hub surface perturbation (top) and comparison of the baseline and optimised non-dimensional pressure distribution on the blade-hub intersection (bottom) for the multiple harmonic case.

Figure 9.14: Detail of the hub boundary layer flow migration for the multiple harmonic case.

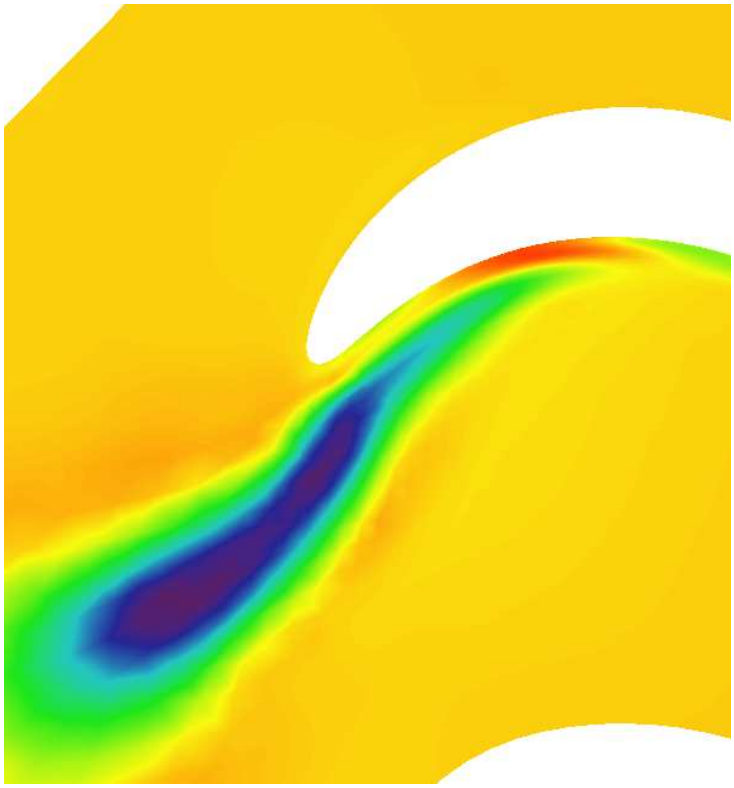


Figure 9.15: Detail of the adjoint solution in the end-wall boundary layer region for the multiple harmonic case.

10. Conclusions and future work

In this thesis, we have presented two strategies to reduce the design time of a turbomachine. One approach to reduce the design time consists in reducing the time spent in the Navier-Stokes equations resolution, by means of methods that accelerate the convergence of the system of equations to its steady state. Among the methods that enhance the convergence rate, the multigrid is a powerful one, since it ensures that the number of operations needed to obtain the solution is $\mathcal{O}(N)$, whatever the number of unknowns N is. It has been demonstrated that the multigrid method enhances the convergence of smooth error waves, which are hardly damped with the sole use of classical iterative schemes, by transferring them into a coarser mesh where their damping is larger. We have also demonstrated the benefits of the multigrid method when solving hyperbolic equations.

The procedure to build the coarser grids needed by the multigrid method has also been addressed for unstructured grids. The coarse grid construction is suitable for edge-based data structure solvers, since the multigrid method is applied upon a baseline solver with that type of data. It consists in an agglomeration method, that fuses neighbouring control volumes to conform an agglomerated mesh with lesser elements. We have presented the algorithms to obtain the necessary data to perform the simulations in coarser grids, paying special attention to the cases where the edge-based agglomeration yields poor quality coarse grid elements.

The multigrid method succeeds in enhancing the convergence rate as long as the high frequency effectively damped by the iterative scheme. Therefore, the iterative scheme should be designed to maximise the damping of these errors. The block-Jacobi preconditioning technique has been successfully used to improve the damping of oscillatory errors in the stretched cells used to mesh boundary layer regions. Even with the use of the block-Jacobi preconditioning, not all the high frequency errors are properly damped. To solve this problem, we have implemented a semi-coarsening technique, that agglomerates the stretched cells that lie close to the walls just in the normal direction. The effectiveness of the preconditioned multigrid method to enhance the convergence rate has been demonstrated for two and three-dimensional cases, hence the

time spent in obtaining a steady solution of the Navier-Stokes equations has been considerably reduced.

Further improvements to enhance the convergence rate could be the use of a line implicit method to implicitly solve the equations in highly stretched cells, which would use the lines that we have constructed for the semi-coarsening technique. Another enhancement could be the use of a GMRES method in conjunction with the multigrid method [14], which is useful especially in the resolution of the adjoint Navier-Stokes equations. It is often the case that the steady solution is not fully converged, typically due to the presence of bubbles next to the walls. In such cases, there are physical eigenvalues of the solution that lie in the imaginary axis of the Fourier space, hence they can not be damped with the use of an explicit iterative method. The use of the GMRES algorithm in conjunction with the iterative method enables the damping of the oscillatory modes, allowing the convergence of the system of adjoint equations.

The design time can also be reduced if the set of design parameters is automatically generated, by means of an optimisation method. We have chosen a gradient-based method, since the number of design parameters is too large to use an evolutionary algorithm. The computation of the gradient of the cost function has been performed with the aid of an adjoint Navier-Stokes solver, which is much less expensive in terms of CPU time than the direct computation of the gradient. We have derived the formulation of the discrete adjoint Navier-Stokes equations and solved them with the use of the convergence acceleration techniques described above. We have demonstrated the equivalence between the linear and adjoint approaches to compute the gradient.

The adjoint solver has been used in conjunction with a gradient-based optimisation method to minimise the secondary losses of a LPT vane using non-axisymmetric end walls. The selected cost function is based on the secondary kinetic energy which has a large sensitivity to variations in the secondary flow pattern. However additional restrictions based on the limitation of the exit swirl angle need to be included to avoid the generation of losses in the downstream rows and provide additional control of separated regions.

A vane of a LPT has been the subject of this optimisation process using two sets of design parameters. The first has a single harmonic perturbation per axial location of the perturbation function, whereas the second has four harmonics. The $SKEH$ has been greatly reduced in both cases, but the latter design has shown better results in terms of both $SKEH$ reduction and flow control close to the end wall. The end wall has also been designed by expert designers finding similar solutions to the ones encountered by the optimiser. However, even in the best cases,

small details of the flow, that are important for the designer, such as small separation regions in the corners, are not prevented by the selected cost function. Future work will concentrate on improving the cost function to fulfill all the designer's criteria in a systematic way. Exploring the use of alternative functions would also be desirable. With the present approach, the cost function and its derivatives are analytically generated and ad-hoc programmed for each case. But the user would like to explore another optimisation targets, e.g., the maximisation of the mass flow. This requires the calculation of another set of cost function derivatives and its implementation, which makes the process impractical from the user's point of view, that has to wait for the programmer to implement the new cost function, and also from the programmer's point of view, that has to code as many functions as the user demands. This could be avoided using a tool to automatically generate the cost functions, by means of a lexical analyser. The user would introduce the cost function into this command interpreter, which would translate it into an efficient programming language. The cost function derivatives could be computed with the use of automatic differentiation techniques. Even though the code resulting from the application of these techniques is not optimal, the computation of the cost function derivatives is not very CPU demanding. These methods would allow the user a greater flexibility to choose arbitrary cost functions.

Bibliography

[1] Allmaras S. Analysis of a Local Matrix Preconditioner for the 2D Navier-Stokes Equations. *AIAA Paper 1993-3330*. 11th AIAA Computational Fluid Dynamics Conference, Orlando, July 1993.

[2] Alonso J., Martinelli L. and Jameson A. Multigrid Unsteady Navier-Stokes Calculations with Aeroelastic Applications. *AIAA Paper 95-0048*. 33rd AIAA Aerospace Sciences Meeting and Exhibit, Reno, January 1995.

[3] Anderson W. and Bonhaus D. Airfoil Design on Unstructured Grids for Turbulent Flows. *AIAA Journal*, vol. 37(2):pages 185–191, 1999.

[4] Anderson W.K. and Venkatakrishnan V. Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. Technical Report TR-97-9, Institute for Computer Applications in Science and Engineering, Jan. 1997.

[5] Arnone A., Liou M. and Povinelli L. Integration of Navier-Stokes Equations Using Dual Time Stepping and a Multigrid Method. *AIAA Journal*, vol. 33(6):pages 985–990, June 1995.

[6] Bakhvalov N. Convergence of a relaxation method with natural constraints on an elliptic operator. *Z. Vycisl. mat. i Mat. Fiz.*, vol. 6:pages 861–885, 1966.

[7] Baldwin B. and Lomax H. Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows. *AIAA Paper 78-257*. 1978.

[8] Bischof C., Carle A., Hovland P., Khademi P. and Mauer A. ADIFOR 2.0 User's guide. Technical Memorandum ANL/MCS-TM-192, Argonne National Laboratory, Jun. 1998.

[9] van den Braembussche R., Alsalihi Z. and Verstraete T. Fast Multidisciplinary Optimization of Turbomachinery Components. *Lecture Series 2004-07: Optimization Methods & Tools for*

Multicritieria/Multidisciplinary Design. Aplications to Aeronautics and Turbomachinery. von Karman Institute for Fluid Dynamics, November 2004.

[10] Brandt A. Multi-Level Adaptative Solutions to Boundary-Value Problems. *Mathematics of Computation*, vol. 31(138):pages 333–390, April 1977.

[11] Briggs W. *A Multigrid Tutorial*. Society for Industrial and Applied Mathematics, 1987.

[12] Burgos M. and Corral R. Application of Phase-Lagged Boundary Conditions to Rotor/Stator interaction. *ASME Paper 2001-GT-0586*. 46th ASME Gas Turbine and Aeroengine Congress, New Orleans, June 2001.

[13] Campobasso M.S., Duta M.C. and Giles M.B. Adjoint Methods for Turbomachinery Design. *ISABE-2001-1055*. International Symposium on Air Breathing Engines, 2001.

[14] Campobasso M.S. and Giles M.B. Effects of Flow Instabilities on the Linear Analysis of Turbomachinery Aeroelasticity. *Journal of Propulsion and Power*, vol. 19:pages 250–259, 2003.

[15] Chorin A. A Numerical Method for Solving Incompressible Viscous Flow Problems. *Journal of Computational Physics*, vol. 2:pages 12–26, 1967.

[16] Contreras J. and Corral R. Quantitative Influence of the Steady Non-Reflecting Boundary Conditions on Blade-to-Blade Computations. *ASME Paper 2000-GT-0515*. 45th ASME Gas Turbine and Aeroengine Congress, Munich, May 2000.

[17] Contreras J., Corral R., Fernández-Castañeda J., Pastor G. and Vasco C. Semiunstructured Grid methods for Turbomachinery Applications. *ASME Paper 2002-GT-30572*. 2002.

[18] Corral R. *Resolución de las Ecuaciones de Euler en régimen transónico mediante métodos multimalla*. Ph.D. thesis, Escuela Técnica Superior de Ingenieros Aeronáuticos, Universidad Politécnica de Madrid, June 1990.

[19] Corral R., Crespo J. and Gisbert F. Parallel Multigrid Unstructured Method for the Solution of the Navier-Stokes Equations. *AIAA Paper 2004-0761*. 42nd AIAA Aerospace Sciences Meeting and Exhibit, Reno, January 2004.

[20] Corral R., Escribano A., Gisbert F., Serrano A. and Vasco C. Validation of a Linear Multigrid Accelerated Unstructured Navier-Stokes Solver for the Computation or Turbine Blades. *AIAA Paper 2003-3326*. 9th AIAA/CEAS Aeroacoustics Conference, May 2003.

[21] Corral R. and Fernández-Castañeda J. Surface Mesh Generation by Means of Steiner Triangulations. *AIAA Journal*, vol. 39(1):pages 176–180, January 2001.

[22] Corral R. and Gisbert F. Non Axisymmetric End-wall Design using an Adjoint Navier-Stokes Solver. *AIAA Paper 2005-4025*. 41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference, Tucson, July 2005.

[23] Corral R. and Gisbert F. Profiled End-wall Desing Using an Adjoint Navier-Stokes Solver. *ASME Paper GT2006-90650*. 51st ASME Gas Turbine and Aeroengine Congress, Barcelona, May 2006.

[24] Courty F., Dervieux A., Koobus B. and Hascoët L. Reverse automatic differentiation for optimum design: from adjoint state assembly to gradient computation. Research Report 4363, INRIA, Sophia-Antipolis, Jan. 2002.

[25] Cusdin P. and Müller J. Automatic Differentiation: Learning to speak AD. Tech. Rep. QUB-SAE-03-05, School of Aeronautical Engineering, Faculty of Engineering, Queen's University, Belfast, Sep. 2003.

[26] Demeulenaere A., Ligout A. and Hirsch C. Application of Multipoint Optimization to the Design of Turbomachinery Blades. *ASME Paper GT2004-53110*. 49th ASME Gas Turbine and Aeroengine Congress, Vienna, June 2004.

[27] Duta M. *The Use of the Adjoint Method for the Minimisation of Forced Vibration in Turbomachinery*. Ph.D. thesis, Oxford University, 2002.

[28] Elliott J. and Peraire J. Practical Three-Dimensional Aerodynamic Design and Optimization Using Unstructured Meshes. *AIAA Journal*, vol. 35(9):pages 1479–1485, 1997.

[29] Fedorenko R. On the speed of convergence of an iteration process. *Z. Vycisl. Mat. i Mat. Fiz.*, vol. 4:pages 559–564, 1964.

[30] Giannakoglou K. Neural Network Assisted Evolutionary Algorithms in Aeronautics and Turbomachinery. *Lecture Series 2004-07: Optimization Methods & Tools for Multicritieria/Multidisciplinary Design. Aplications to Aeronautics and Turbomachinery*. von Karman Institute for Fluid Dynamics, November 2004.

[31] Giannakoglou K., Papadimitrou D. and Kampolis I. Coupling Evolutionary Algorithms, Surrogate Models and Adjoint Methods in Inverse Design and Optimization Problems.

Lecture Series 2004-07: Optimization Methods & Tools for Multicritieria/Multidisciplinary Design. Aplications to Aeronautics and Turbomachinery. von Karman Institute for Fluid Dynamics, November 2004.

[32] Giles M. Non-reflecting Boundary Conditions for Euler Equations. *AIAA Journal*, vol. 28(12):pages 2050–2057, 1990.

[33] Giles M. On the Use of Runge-Kutta Time-marching and Multigrid for the Solution of Steady Adjoint Equations. Technical Report NA00/10, Oxford University Computing Laboratory, Jun. 2000.

[34] Giles M., Duta M. and Müller J. Adjoint Code Developments Using the Exact Discrete Approach. *AIAA Paper 2001-2596*. 15th Computational Fluid Dynamics Conference, Anaheim, California, Jun. 2001.

[35] Giles M., Duta M., Müller J. and Pierce N. Algorithm Developments for Discrete Adjoint Methods. *AIAA Journal*, vol. 122(2):pages 198–205, Feb. 2003.

[36] Giles M.B. and Pierce N.A. An Introduction to the Adjoint Approach to Design. *Flow, Turbulence and Combustion*, vol. 65:pages 393–415, 2000.

[37] Gómez R. *Una Estructura de Datos basada en Aristas para la Resolución de las Ecuaciones de Navier-Stokes*. Ph.D. thesis, Escuela Técnica Superior de Ingenieros Aeronáuticos, Universidad Politécnica de Madrid, Noviembre 2000.

[38] Hackbusch W. On the multi-grid method applied to difference equations. *Journal of Computing*, vol. 20:pages 291–306, 1978.

[39] Harten A. High resolution schemes for hyperbolic conservation laws. *Journal of Computational Physics*, vol. 49:pages 357–393, 1983.

[40] Harvey N., Rose M., Shahpar S., Taylor M., Hartland J. and Gregory-Smith D. Non-Axisymmetric Turbine End Wall Design: Part I Three-Dimensional Design System. *Journal of Turbomachinery*, vol. 122:pages 278–285, 1999.

[41] Jameson A. Solution of the Euler Equations For Two Dimensional Transonic Flow by a Multigrid Method. *Applied Mathematics and Computation*, vol. 13:pages 327–356, 1983.

[42] Jameson A. Aerodynamic design via control theory. *Journal of Scientific Computing*, vol. 3:pages 233–260, 1988.

[43] Jameson A. Time Dependent Calculations Using Multigrid with Applications to Unsteady Flows past Airfoils and Wings. *AIAA Paper 1991-1596*. 10th Computational Fluid Dynamics Conference, Honolulu, June 1991.

[44] Jameson A. Optimum Transonic Wing Design using Control Theory. *Symposium Transsonicum IV*. International Union of Theoretical and Applied Mechanics, DLR, Gottingen, Germany, Sep. 2002.

[45] Jameson A. Aerodynamic Shape Optimization Using the Adjoint Method. Tech. rep., Lectures at the Von Karman Institute, Feb. 2003.

[46] Jameson A., Martinelli L. and Grasso F. A Multistage Multigrid Method for the Compressible Navier Stokes Equations. *Notes on Numerical Fluid Mechanics*, vol. 18:pages 123–128, 1987.

[47] Jameson A., Pierce N. and Martinelli L. Optimum Aerodynamic Design using the Navier Stokes Equation. *Theoretical and Computational Fluid Dynamics*, vol. 10:pages 213–237, 1998.

[48] Jameson A., Schmidt W. and Turkel E. Numerical solution of the Euler equations by finite volume methods using Runge-Kutta time stepping schemes. *AIAA Paper 81-1259*. 14th Fluid and Plasma Dynamic Conference, Palo Alto, June 1981.

[49] Jameson A., Shankaran S., Martinelli L. and Haimes B. Aerodynamic Shape Optimization of Complete Aircraft Configurations using Unstructured Grids. *AIAA Paper 2004-0533*. 42nd AIAA Aerospace Science Meeting, Reno, Nevada, January 2004.

[50] Kim S., Alonso J. and Jameson A. Design Optimization of High Lift Configurations Using a Viscous Continuous Adjoint Method. *AIAA Paper 2002-0844*. 40th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, Jan. 2002.

[51] van Leer B., Lee W. and Roe P. Characteristic Time-stepping or Local Preconditioning of the Euler Equations. *AIAA Paper 1991-1552*. 10th Computational Fluid Dynamics Conference, Honolulu, June 1991.

[52] Liu F. and Zheng X. A Strongly Coupled Time-Marching Method for solving the Navier-Stokes equations and $k-\omega$ Turbulence Model Equations with Multigrid. *Journal of Computational Physics*, vol. 128:pages 289–300, 1996.

[53] Luo H., Baum J. and Löhner R. Edge-Based Finite Element Scheme for the Euler Equations. *AIAA Journal*, vol. 32:pages 1183–1190, 1994.

[54] Martel C. Nonlinear Constrained Optimization of Turbomachinery Problems. Tech. Rep. ETSIA FM/00-01, Escuela Técnica Superior de Ingenieros Aeronáuticos, December 2000.

[55] Martinelli L. *Calculations of viscous flow with a multigrid method*. Ph.D. thesis, Princeton University, 1987.

[56] Mavriplis D. *Solution of the Two-Dimensional Euler Equations on Unstructured Triangular Meshes*. Ph.D. thesis, Mechanical and Aerospace Engineering Department, Princeton University, 1987.

[57] Mavriplis D. Multigrid Techniques for Unstructured Meshes. Lecture series, von Karman Institute for Fluid Dynamics, March 1995.

[58] Mavriplis D. Directional Coarsening and Smoothing for Anisotropic Navier-Stokes Problems. *Electronic Transactions on Numerical Analysis*, vol. 6:pages 182–197, December 1997.

[59] Mavriplis D. On Convergence Acceleration Techniques for Unstructured Meshes. *AIAA Paper 98-2966*. 29th AIAA Fluid Dynamics Conference, Albuquerque, NM, June 1998.

[60] Mavriplis D. Directional Agglomeration Multigrid Techniques for High Reynolds Number Viscous Flows. *AIAA Journal*, vol. 37(10):pages 1222–1230, October 1999.

[61] Mavriplis D. and Jameson A. Multigrid Solution of the Navier-Stokes Equations on Triangular Meshes. *AIAA Journal*, vol. 28:pages 1415–1425, 1990.

[62] Mavriplis D. and Venkatakrishnan V. A Unified Multigrid Solver for the Navier-Stokes Equations on Mixed Element Meshes. Tech. Rep. ICASE Report No. 95-53, Institute for Computer Applications in Science and Engineering, July 1995.

[63] Moinier P. *Algorithm Developments for an Unstructured Viscous Flow Solver*. Ph.D. thesis, University of Oxford, 1999.

[64] Mulder W. A new multigrid approach to convection problems. *Journal of Computational Physics*, vol. 83:pages 303–323, 1989.

[65] Müller J. Coarsening 3-D Hybrid Meshes for Multigrid Methods. *Copper Mountain Conference on Multigrid Methods*. 1999.

[66] Müller J. and Giles M. Edge-based multigrid schemes for hybrid grids. *6th ICFD Conference on Numerical Methods for Fluid Dynamics*. 1998.

[67] Nadarajah S. *The Discrete Adjoint Approach to Aerodynamic Shape Optimization*. Ph.D. thesis, Department of Aeronautics and Astronautics, Stanford University, 2003.

[68] Nadarajah S. and Jameson A. A Comparison of the Continous and Discrete Adjoint Approach to Automatic Aerodynamic Optimization. *AIAA Paper 2000-0667*. 2000.

[69] Ni R. A Multiple Grid Scheme for Solving the Euler Equations. *AIAA 5th Computational Fluid Dynamics Conference*, pages 257–264. 1981.

[70] Nicolaides R. On the l^2 convergence of an algorithm for solving finite element equations. *Mathematics of Computation*, vol. 31:pages 892–906, 1977.

[71] Nielsen E., Lu J., Park M. and Darmofal D. An Exact Dual Adjoint Solution Method for Turbulent Flows on Unstructured Grids. *AIAA Paper 2003-0272*. 41st AIAA Aerospace Sciences Meeting and Exhibit, Reno, Jan 2003.

[72] Pierce N. *Preconditioned Multigrid Methods for Compressible Flow Calculations on Stretched Meshes*. Ph.D. thesis, University of Oxford, 1997.

[73] Pierce N. and Giles M. Preconditioning on Stretched Meshes. Technical Report NA95/10, Oxford University Computing Laboratory, June 1995.

[74] Pierce N. and Giles M. Preconditioned Multigrid Methods for Compressible Flow Calculations on Stretched Meshes. *Journal of Computational Physics*, vol. 136:pages 425–445, May 1997.

[75] Pierce N., Giles M., Jameson A. and Martinelli L. Accelerating Three-Dimensional Navier-Stokes Calculations. *AIAA Paper 1997-1953*. AIAA 13th Computational Fluid Dynamics Conference, Snowmass, Colorado, June 1997.

[76] Pironneau O. On optimum design in fluid mechanics. *Journal of Fluid Mechanics*, vol. 64:pages 97–110, 1974.

[77] Prasad A. Calculation of the Mixed-Out State in Turbomachine Flows. *Journal of Turbomachinery*, vol. 127:pages 564–572, July 2005.

[78] Roe P. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, vol. 43:pages 357–372, 1981.

[79] Rose M. and Martin G. Non-Axisymmetric Endwall Profiling in the HP NGVs of an Axial Flow Gas Turbine. *ASME Paper 94-GT-249*. International Gas Turbine and Aeroengine Congress, 1994.

[80] Rottmeier F. Steady-State and Controlled Vibration Experiments at EPFL. Tech. Rep. ADTB-EPFL-3003, EPFL, September 1999.

[81] Torre D., Vázquez R., de la Rosa Blanco E. and Hodson H.P. A New Alternative for Reduction of Secondary Flows in Low Pressure Turbines. *ASME Paper 2006-GT-91002*. 51st ASME Gas Turbine and Aeroengine Congress, Barcelona, May 2006.

[82] Turkel E. Preconditioned Methods for Solving the Incompressible and Low Speed Compressible Equations. *Journal of Computational Physics*, vol. 72:pages 277–298, 1987.

[83] Turkel E. Preconditioning Techniques in Computational Fluid Dynamics. *Annual Review of Fluid Mechanics*, vol. 31:pages 385–416, 1999.

[84] Turkel E., Fiterman A. and van Leer B. *Computing the Future: Advances and Prospects for Computational Aerodynamics*, chap. Preconditioning and the Limit to the Incompressible Flow equations for finite difference schemes, pages 215–234. John Wiley & Sons, 1994.

[85] Waltz J. and Löhner R. A Grid Coarsening Algorithm for Unstructured Multigrid Applications. *AIAA Paper 2000-0925*. 38th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Jan 2000.

[86] Wilcox D.C. *Turbulence Modeling for CFD*. DCW Industries, Inc., 1998.

[87] Wood J., Strasizar T. and Hathaway M. Test Case E/CA-6, Subsonic Turbine Cascade T106. *Test Cases for Computation of Internal Flows*. AGARD-AR-275, July 1990.

A. Formulation of the block-Jacobi Preconditioning Matrices

In Eq. (5.9), the matrices $|A_{ik}|$ and B_{ik} are

$$|A_{ik}| = T_{ik} \, |\Lambda_{ik}| \, T_{ik}^{-1} \tag{A.1}$$

being

$$|\Lambda_{ik}| = \begin{bmatrix} |v_n| & 0 & 0 & 0 & 0 \\ 0 & |v_n| & 0 & 0 & 0 \\ 0 & 0 & |v_n| & 0 & 0 \\ 0 & 0 & 0 & |v_n + c| & 0 \\ 0 & 0 & 0 & 0 & |v_n - c| \end{bmatrix}, \tag{A.2}$$

The eigenvector matrices are

$$T_{ik} = \begin{bmatrix} n_x & n_y & n_z & 1 & 1 \\ un_x & un_y - cn_z & un_z + cn_y & u + cn_x & u - cn_x \\ vn_x + cn_z & vn_y & vn_z - cn_x & v + cn_y & v - cn_y \\ wn_x - cn_y & wn_y + cn_x & wn_z & w + cn_z & w - cn_z \\ \frac{q^2}{2} n_x + c\,(vn_z - wn_y) & \frac{q^2}{2} n_y + c\,(wn_x - un_z) & \frac{q^2}{2} n_z + c\,(un_y - vn_x) & H + cv_n & H - cv_n \end{bmatrix},$$

and its inverse

$$T_{ik}^{-1} = \frac{1}{c^2} \begin{bmatrix} T_1 & T_2 & T_3 & T_4 & T_5 \end{bmatrix}$$

$$
T_1 = \begin{Bmatrix}
\left(c^2 - (\gamma-1)\frac{q^2}{2}\right) n_x + c\,(wn_y - vn_z) \\
\left(c^2 - (\gamma-1)\frac{q^2}{2}\right) n_y + c\,(un_z - wn_x) \\
\left(c^2 - (\gamma-1)\frac{q^2}{2}\right) n_z + c\,(vn_x - un_y) \\
\frac{1}{2}\left((\gamma-1)\frac{q^2}{2} - cv_n\right) \\
\frac{1}{2}\left((\gamma-1)\frac{q^2}{2} + cv_n\right)
\end{Bmatrix}
$$

$$
T_2 = \begin{Bmatrix}
(\gamma-1)\,un_x \\
(\gamma-1)\,un_y - cn_z \\
(\gamma-1)\,un_z + cn_y \\
-\frac{1}{2}\left((\gamma-1)\,u - cn_x\right) \\
-\frac{1}{2}\left((\gamma-1)\,u + cn_x\right)
\end{Bmatrix}
$$

$$
T_3 = \begin{Bmatrix}
(\gamma-1)\,vn_x + cn_z \\
(\gamma-1)\,vn_y \\
(\gamma-1)\,vn_z - cn_x \\
-\frac{1}{2}\left((\gamma-1)\,v - cn_y\right) \\
-\frac{1}{2}\left((\gamma-1)\,v + cn_y\right)
\end{Bmatrix}
$$

$$
T_4 = \begin{Bmatrix}
(\gamma-1)\,wn_x - cn_y \\
(\gamma-1)\,wn_y + cn_x \\
(\gamma-1)\,wn_z \\
-\frac{1}{2}\left((\gamma-1)\,w - cn_z\right) \\
-\frac{1}{2}\left((\gamma-1)\,w + cn_z\right)
\end{Bmatrix}
$$

$$
T_5 = \begin{Bmatrix}
-(\gamma-1)\,n_x \\
-(\gamma-1)\,n_y \\
-(\gamma-1)\,n_z \\
\frac{1}{2}(\gamma-1) \\
\frac{1}{2}(\gamma-1)
\end{Bmatrix}
$$

and

$$
B_{ik} = \begin{bmatrix} B_1 & B_2 & B_3 & B_4 & B_5 \end{bmatrix}
$$

with

$$
B_1 = \left\{ \begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ -\dfrac{\gamma p}{\rho^2 (\gamma - 1)} \left(\dfrac{\mu}{Pr} \right)_{eq} \mathbf{n} \cdot \mathbf{l} \end{array} \right\}
$$

$$
B_2 = \left\{ \begin{array}{c} 0 \\ \mu_{eq} \left(\mathbf{n} \cdot \mathbf{l} + \dfrac{1}{3} n_x l_x \right) \\ \mu_{eq} \left(-\dfrac{2}{3} n_y l_x + n_x l_y \right) \\ \mu_{eq} \left(-\dfrac{2}{3} n_z l_x + n_x l_z \right) \\ \mu_{eq} \left[u \cdot \left(\mathbf{n} \cdot \mathbf{l} + \dfrac{1}{3} n_x l_x \right) + v \cdot \left(-\dfrac{2}{3} n_y l_x + n_x l_y \right) + w \cdot \left(-\dfrac{2}{3} n_z l_x + n_x l_z \right) \right] \end{array} \right\}
$$

$$
B_3 = \left\{ \begin{array}{c} 0 \\ \mu_{eq} \left(-\dfrac{2}{3} n_x l_y + n_y l_x \right) \\ \mu_{eq} \left(\mathbf{n} \cdot \mathbf{l} + \dfrac{1}{3} n_y l_y \right) \\ \mu_{eq} \left(-\dfrac{2}{3} n_z l_y + n_y l_z \right) \\ \mu_{eq} \left[u \cdot \left(-\dfrac{2}{3} n_x l_y + n_y l_x \right) + v \cdot \left(\mathbf{n} \cdot \mathbf{l} + \dfrac{1}{3} n_y l_y \right) + w \cdot \left(-\dfrac{2}{3} n_z l_y + n_y l_z \right) \right] \end{array} \right\}
$$

$$
B_4 = \left\{ \begin{array}{c} 0 \\ \mu_{eq} \left(-\dfrac{2}{3} n_x l_z + n_z l_x \right) \\ \mu_{eq} \left(-\dfrac{2}{3} n_y l_z + n_z l_y \right) \\ \mu_{eq} \left(\mathbf{n} \cdot \mathbf{l} + \dfrac{1}{3} n_z l_z \right) \\ \mu_{eq} \left[u \cdot \left(-\dfrac{2}{3} n_x l_z + n_z l_x \right) + v \cdot \left(-\dfrac{2}{3} n_y l_z + n_z l_y \right) + w \cdot \left(\mathbf{n} \cdot \mathbf{l} + \dfrac{1}{3} n_z l_z \right) \right] \end{array} \right\}
$$

$$
B_5 = \left\{ \begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ \dfrac{\gamma}{(\gamma - 1) \rho} \left(\dfrac{\mu}{Pr} \right)_{eq} \mathbf{n} \cdot \mathbf{l} \end{array} \right\}
$$

In all these values we have defined the following variables:

$$\mathbf{n} = \left\{ \begin{array}{ccc} n_x & n_y & n_z \end{array} \right\} = \frac{1}{|\sigma_{ik}|} \left\{ \begin{array}{ccc} \sigma_{ik}^{(x)} & \sigma_{ik}^{(y)} & \sigma_{ik}^{(z)} \end{array} \right\}$$

$$\mathbf{l} = \left\{ \begin{array}{ccc} l_x & l_y & l_z \end{array} \right\} = \frac{1}{|\mathbf{x}_k - \mathbf{x}_i|} \left\{ \begin{array}{ccc} x_k - x_i & y_k - y_i & z_k - z_i \end{array} \right\}$$

$H = c^2/(\gamma - 1) + q^2/2$ is the stagnation enthalpy, c the speed of sound and $q^2 = \mathbf{v} \cdot \mathbf{v}$. The variables in all the matrices are the mean between the two nodes of the edge ik.

When using the low Mach number preconditioning together with the block-Jacobi, the eigenvalues of the problem are modified, and the new matrices used in Eq. 5.23 are

$$\widetilde{M}_{ik}\Gamma_{ik}^{-1} \left|\Gamma \widetilde{A}_{ik}\right| \widetilde{M}_{ik}^{-1} = \widetilde{M}_{ik}\Gamma_{ik}^{-1} G_{ik} \left|\widetilde{\Lambda}_{ik}\right| G_{ik}^{-1} \widetilde{M}_{ik}^{-1},$$

where

$$\left|\widetilde{\Lambda}_{ik}\right| = \begin{bmatrix} |v_n| & 0 & 0 & 0 & 0 \\ 0 & |v_n| & 0 & 0 & 0 \\ 0 & 0 & |v_n| & 0 & 0 \\ 0 & 0 & 0 & \left|\frac{1}{2}(1+\varepsilon)\, v_n + \frac{\tau}{2}\right| & 0 \\ 0 & 0 & 0 & 0 & \left|\frac{1}{2}(1+\varepsilon)\, v_n - \frac{\tau}{2}\right| \end{bmatrix}$$

is the modified absolute eigenvalue matrix, and

$$\widetilde{M}_{ik}\Gamma_{ik}^{-1} G_{ik} = \left[\begin{array}{ccccc} L_1 & L_2 & L_3 & L_4 & L_5 \end{array} \right],$$

being

$$L_1 = \left\{ \begin{array}{c} n_x \\ u n_x \\ v n_x + c n_z \\ w n_x - c n_y \\ \frac{q^2}{2} n_x + c\,(v n_z - w n_y) \end{array} \right\}$$

$$
\begin{aligned}
L_2 &= \left\{ \begin{array}{c} n_y \\ un_y - cn_z \\ vn_y \\ wn_y + cn_x \\ \frac{q^2}{2} n_y + c\left(wn_x - un_z\right) \end{array} \right\} \\
L_3 &= \left\{ \begin{array}{c} n_z \\ un_z + cn_y \\ vn_z - cn_x \\ wn_z \\ \frac{q^2}{2} n_z + c\left(un_y - vn_x\right) \end{array} \right\} \\
L_4 &= \left\{ \begin{array}{c} \frac{s^-}{2c\varepsilon} \\ \frac{us^- + 2c^2 n_x \varepsilon}{2c\varepsilon} \\ \frac{vs^- + 2c^2 n_y \varepsilon}{2c\varepsilon} \\ \frac{ws^- + 2c^2 n_z \varepsilon}{2c\varepsilon} \\ \frac{Hs^- + 2c^2 v_n \varepsilon}{2c\varepsilon} \end{array} \right\} \\
L_5 &= \left\{ \begin{array}{c} \frac{s^+}{2c\varepsilon} \\ \frac{us^+ - 2c^2 n_x \varepsilon}{2c\varepsilon} \\ \frac{vs^+ - 2c^2 n_y \varepsilon}{2c\varepsilon} \\ \frac{ws^+ - 2c^2 n_z \varepsilon}{2c\varepsilon} \\ \frac{Hs^+ - 2c^2 v_n \varepsilon}{2c\varepsilon} \end{array} \right\}
\end{aligned}
$$

and

$$
G^{-1}\widetilde{M}^{-1} = \left[\begin{array}{ccccc} R_1 & R_2 & R_3 & R_4 & R_5 \end{array} \right],
$$

$$
R_1 = \left\{ \begin{array}{c} \frac{1}{c^2}\left[\left(c^2 - (\gamma-1)\frac{q^2}{2}\right) n_x + c\left(wn_y - vn_z\right)\right] \\ \frac{1}{c^2}\left[\left(c^2 - (\gamma-1)\frac{q^2}{2}\right) n_y + c\left(un_z - wn_x\right)\right] \\ \frac{1}{c^2}\left[\left(c^2 - (\gamma-1)\frac{q^2}{2}\right) n_z + c\left(vn_x - un_y\right)\right] \\ \frac{1}{c\tau}\left((\gamma-1)\frac{q^2}{2} - \frac{s^+ v_n}{2}\right) \\ \frac{1}{2c\tau}\left((\gamma-1)\frac{q^2}{2} + \frac{s^- v_n}{2}\right) \end{array} \right\}
$$

$$
R_2 = \left\{ \begin{array}{c} \frac{1}{c^2}(\gamma-1)\,un_x \\ \frac{1}{c^2}\left[(\gamma-1)\,un_y - cn_z\right] \\ \frac{1}{c^2}\left[(\gamma-1)\,un_z + cn_y\right] \\ -\frac{1}{c\tau}\left((\gamma-1)\,u - \frac{s^+ n_x}{2}\right) \\ -\frac{1}{c\tau}\left((\gamma-1)\,u + \frac{s^- n_x}{2}\right) \end{array} \right\}
$$

$$
R_3 = \left\{ \begin{array}{c} \frac{1}{c^2}\left[(\gamma-1)\,vn_x + cn_z\right] \\ \frac{1}{c^2}(\gamma-1)\,vn_y \\ \frac{1}{c^2}\left[(\gamma-1)\,vn_z - cn_x\right] \\ -\frac{1}{c\tau}\left((\gamma-1)\,v - \frac{s^+ n_y}{2}\right) \\ -\frac{1}{c\tau}\left((\gamma-1)\,v + \frac{s^- n_y}{2}\right) \end{array} \right\}
$$

$$
R_4 = \left\{ \begin{array}{c} \frac{1}{c^2}\left[(\gamma-1)\,wn_x - cn_y\right] \\ \frac{1}{c^2}\left[(\gamma-1)\,wn_y + cn_x\right] \\ \frac{1}{c^2}(\gamma-1)\,wn_z \\ -\frac{1}{c\tau}\left((\gamma-1)\,w - \frac{s^+ n_z}{2}\right) \\ -\frac{1}{c\tau}\left((\gamma-1)\,w + \frac{s^- n_z}{2}\right) \end{array} \right\}
$$

$$
R_5 = \left\{ \begin{array}{c} -\frac{1}{c^2}(\gamma-1)\,n_x \\ -\frac{1}{c^2}(\gamma-1)\,n_y \\ -\frac{1}{c^2}(\gamma-1)\,n_z \\ \frac{1}{c\tau}(\gamma-1) \\ \frac{1}{c\tau}(\gamma-1) \end{array} \right\}
$$

being

$$
\begin{aligned} \tau &= \sqrt{(1-\varepsilon)^2 v_n^2 + 4\varepsilon c^2} \\ s^+ &= \tau + (1-\varepsilon)\,v_n \\ s^- &= \tau - (1-\varepsilon)\,v_n \end{aligned}
$$

B. Adjoint Navier-Stokes equations. Implementation and resolution

The adjoint discrete Navier-Stokes equations are derived from the discrete linearised Navier-Stokes equations in this appendix. First, the finite volume discretisation of the Navier-Stokes equations written in Eq. (2.7), is used to linearise the discrete equations. Then, the adjoint operator is obtained from its linear counterpart. Once the adjoint Navier-Stokes equations are discretised, some resolution techniques are also developed. Even though the methods used for the resolution of the linear system of equations are also valid for the adjoint problem, because the linear and adjoint problems share the same eigenvalues, the adjoint counterparts of the linear resolution methods provide a valid approach to debug the implementation of the adjoint discrete equations. Thus, when the adjoint resolution methods are used, it is ensured that the linear and the adjoint formulations yield the same value of the functional when the same number of iterations have been performed [33].

B.1. Adjoint inviscid fluxes

The jacobian of the inviscid fluxes results from the derivation of the fluxes of Eq. (2.2) with respect to the conservative variables:

$$A_U = \frac{\partial \left(\mathbf{F}_c \cdot \mathbf{n} \right)}{\partial \mathbf{U}} = \left[\begin{array}{ccccc} \mathbf{F}_{c1} & \mathbf{F}_{c2} & \mathbf{F}_{c3} & \mathbf{F}_{c4} & \mathbf{F}_{c5} \end{array} \right]$$

$$\mathbf{F}_{c1} = \begin{bmatrix} 0 \\ -u \cdot v_n + n_x (\gamma - 1) \dfrac{q^2}{2} \\ -v \cdot v_n + n_y (\gamma - 1) \dfrac{q^2}{2} \\ -w \cdot v_n + n_z (\gamma - 1) \dfrac{q^2}{2} \\ v_n \left(-\gamma E + (\gamma - 1) q^2 \right) \end{bmatrix}$$

$$\mathbf{F}_{c2} = \begin{bmatrix} n_x \\ (3 - \gamma) u n_x + v n_y + w n_z \\ v n_x - (\gamma - 1) u n_y \\ w n_x - (\gamma - 1) u n_z \\ \gamma E n_x - (\gamma - 1) \left(v_n \cdot u + \dfrac{q^2}{2} n_x \right) \end{bmatrix}$$

$$\mathbf{F}_{c3} = \begin{bmatrix} n_y \\ u n_y - (\gamma - 1) v n_x \\ u n_x + (3 - \gamma) v n_y + w n_z \\ w n_y - (\gamma - 1) v n_z \\ \gamma E n_y - (\gamma - 1) \left(v_n v + \dfrac{q^2}{2} n_y \right) \end{bmatrix}$$

$$\mathbf{F}_{c4} = \begin{bmatrix} n_z \\ u n_z - (\gamma - 1) w n_x \\ v n_z - (\gamma - 1) w n_y \\ u n_x + v n_y + (3 - \gamma) w n_z \\ \gamma E n_z - (\gamma - 1) \left(v_n w + \dfrac{q^2}{2} n_z \right) \end{bmatrix}$$

$$\mathbf{F}_{c5} = \begin{bmatrix} 0 \\ (\gamma - 1) n_x \\ (\gamma - 1) n_y \\ (\gamma - 1) n_z \\ \gamma v_n \end{bmatrix}$$

The formulation is simpler when expressed in primitive variables $\mathbf{V} = (\rho\ u\ v\ w\ p)$, hence a change of coordinates represented by a matrix M will be used. Thus $A_U = A_V M^{-1}$, where

$$M^{-1} = \frac{\partial \mathbf{V}}{\partial \mathbf{U}} = \frac{\partial (\rho\ u\ v\ w\ p)}{\partial (\rho\ \rho u\ \rho v\ \rho w\ \rho E)} =$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -\frac{u}{\rho} & \frac{1}{\rho} & 0 & 0 & 0 \\ -\frac{v}{\rho} & 0 & \frac{1}{\rho} & 0 & 0 \\ -\frac{w}{\rho} & 0 & 0 & \frac{1}{\rho} & 0 \\ (\gamma-1)\frac{q^2}{2} & -(\gamma-1)u & -(\gamma-1)v & -(\gamma-1)w & \gamma-1 \end{bmatrix}$$

and

$$A_V = \frac{\partial(\mathbf{F}_c \cdot \mathbf{n})}{\partial \mathbf{V}} = \begin{bmatrix} \mathbf{F}_{p1} & \mathbf{F}_{p2} & \mathbf{F}_{p3} & \mathbf{F}_{p4} & \mathbf{F}_{p5} \end{bmatrix},$$

$$\mathbf{F}_{p1} = \begin{bmatrix} v_n \\ u \cdot v_n \\ v \cdot v_n \\ w \cdot v_n \\ v_n \dfrac{q^2}{2} \end{bmatrix}$$

$$\mathbf{F}_{p2} = \begin{bmatrix} \rho n_x \\ \rho v_n + \rho u \cdot n_x \\ \rho v \cdot n_x \\ \rho w \cdot n_x \\ \rho u \cdot v_n + \rho H n_x \end{bmatrix}$$

$$\mathbf{F}_{p3} = \begin{bmatrix} \rho n_y \\ \rho u \cdot n_y \\ \rho v_n + \rho v \cdot n_y \\ \rho w \cdot n_y \\ \rho v \cdot v_n + \rho H n_y \end{bmatrix}$$

$$\mathbf{F}_{p4} = \begin{bmatrix} \rho n_z \\ \rho u \cdot n_z \\ \rho v \cdot n_z \\ \rho v_n + \rho w \cdot n_z \\ \rho w \cdot v_n + \rho H n_z \end{bmatrix}$$

$$\mathbf{F}_{p5} = \begin{bmatrix} 0 \\ n_x \\ n_y \\ n_z \\ \dfrac{\gamma v_n}{\gamma - 1} \end{bmatrix}$$

where $H = c^2/(\gamma - 1) + q^2/2$ is the stagnation enthalpy, c is the speed of sound and $q^2 = u^2 + v^2 + w^2$. When the flux is calculated, the edge contribution is, according to Eq. (2.7):

$$\mathbf{F}_{ij} = \frac{1}{2}\left(\mathbf{F}_j + \mathbf{F}_i\right)\mathbf{n}_{ij}\sigma_{ij}$$

which linearised yields

$$\mathbf{LF}^c_{ij} = \frac{\partial \mathbf{F}_{ij}}{\partial \boldsymbol{\varphi}_k} = \frac{1}{2}\left(A_{Vi}M_i^{-1}\frac{\partial \mathbf{U}_i}{\partial \boldsymbol{\varphi}_k} + A_{Vj}M_j^{-1}\frac{\partial \mathbf{U}_j}{\partial \boldsymbol{\varphi}_k}\right)\sigma_{ij}$$

This term is associated to an edge, and its contribution is splitted onto the nodes that belong to that edge. That yields a contribution to the vector of fluxes that can be written as a product of the system matrix and the linearised flow variables:

$$\begin{bmatrix} 0 \\ \vdots \\ \mathbf{LF}^c_i \\ \vdots \\ \mathbf{LF}^c_j \\ \vdots \\ 0 \end{bmatrix} = \frac{\sigma_{ij}}{2}\begin{bmatrix} 0 & & \cdots & & 0 \\ \vdots & \ddots & & & \vdots \\ 0 & \cdots & A_{Vi} & \cdots & A_{Vj} & \cdots & 0 \\ \vdots & & & 0 & & & \vdots \\ 0 & \cdots & -A_{Vi} & \cdots & -A_{Vj} & \cdots & 0 \\ \vdots & & & & & \ddots & \vdots \\ 0 & & \cdots & & & & 0 \end{bmatrix} M^{-1} \begin{bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_i \\ \vdots \\ \mathbf{u}_j \\ \vdots \\ \mathbf{u}_N \end{bmatrix}, \tag{B.1}$$

being

$$M^{-1} = \begin{bmatrix} M_1^{-1} & & & \cdots & & & 0 \\ \vdots & \ddots & & & & & \vdots \\ 0 & \cdots & M_i^{-1} & \cdots & & \cdots & 0 \\ \vdots & & & & & & \vdots \\ 0 & \cdots & & \cdots & M_j^{-1} & \cdots & 0 \\ \vdots & & & & & \ddots & \vdots \\ 0 & & & \cdots & & & M_N^{-1} \end{bmatrix}$$

and $\mathbf{u}_i = \partial \mathbf{U}_i / \partial \boldsymbol{\varphi}_k$. From now on, Eq. (B.1) and similar expressions will be expressed just as

$$\begin{bmatrix} \mathbf{LF}_i^c \\ \mathbf{LF}_j^c \end{bmatrix} = \frac{\sigma_{ij}}{2} \begin{bmatrix} A_{Vi} & A_{Vj} \\ -A_{Vi} & -A_{Vj} \end{bmatrix} M^{-1} \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_j \end{bmatrix},$$

where the null terms of the matrix and vectors have been omitted.

When transposing the matrix we obtain the adjoint inviscid fluxes, that are expressed as:

$$\begin{bmatrix} \mathbf{AF}_i^c \\ \mathbf{AF}_j^c \end{bmatrix} = \frac{\sigma_{ij}}{2} \left(M^{-1}\right)^T \begin{bmatrix} A_{Vi}^T & -A_{Vi}^T \\ A_{Vj}^T & -A_{Vj}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_i \\ \boldsymbol{\psi}_j \end{bmatrix},$$

therefore the adjoint contribution is, for each edge:

$$\begin{aligned} \mathbf{AF}_i^c &= \frac{1}{2} \left(M_i^{-1}\right)^T A_{Vi}^T \left(\boldsymbol{\psi}_i - \boldsymbol{\psi}_j\right) \sigma_{ij} \\ \mathbf{AF}_j^c &= \frac{1}{2} \left(M_j^{-1}\right)^T A_{Vj}^T \left(\boldsymbol{\psi}_i - \boldsymbol{\psi}_j\right) \sigma_{ij} \end{aligned}$$

The product $A_V^T \Delta\boldsymbol{\psi}$ yields

$$A_V^T \Delta\boldsymbol{\psi} = \begin{bmatrix} v_n \left(\Delta\widetilde{\psi}_1 + \mathbf{v} \cdot \Delta\widetilde{\boldsymbol{\psi}}_{234}\right) \\ \rho \left(\Delta\widetilde{\psi}_1 + \mathbf{v} \cdot \Delta\widetilde{\boldsymbol{\psi}}_{234} + \dfrac{c^2}{\gamma - 1} \Delta\psi_5\right) n_x + \rho v_n \Delta\widetilde{\psi}_2 \\ \rho \left(\Delta\widetilde{\psi}_1 + \mathbf{v} \cdot \Delta\widetilde{\boldsymbol{\psi}}_{234} + \dfrac{c^2}{\gamma - 1} \Delta\psi_5\right) n_y + \rho v_n \Delta\widetilde{\psi}_3 \\ \rho \left(\Delta\widetilde{\psi}_1 + \mathbf{v} \cdot \Delta\widetilde{\boldsymbol{\psi}}_{234} + \dfrac{c^2}{\gamma - 1} \Delta\psi_5\right) n_z + \rho v_n \Delta\widetilde{\psi}_4 \\ \Delta\widetilde{\boldsymbol{\psi}}_{234} \cdot \mathbf{n} + \dfrac{v_n}{\gamma - 1} \Delta\psi_5 \end{bmatrix},$$

where

$$\begin{aligned}
\Delta\boldsymbol{\psi} &= \boldsymbol{\psi}_i - \boldsymbol{\psi}_j \\
\Delta\widetilde{\psi}_1 &= \Delta\psi_1 - \frac{q^2}{2}\psi_5 \\
\Delta\widetilde{\boldsymbol{\psi}}_{234} &= \left\{ \begin{array}{c} \Delta\psi_2 + u\Delta\psi_5 \\ \Delta\psi_3 + v\Delta\psi_5 \\ \Delta\psi_4 + w\Delta\psi_5 \end{array} \right\}
\end{aligned}$$

B.2. Adjoint artificial viscosity fluxes

Since the complete linearisation of the numerical diffusion formulation is a painful task and increases the cost of computing such terms very much, we have chosen an alternative approach. The artificial dissipation fluxes for an edge ij are expressed as the product

$$|A_{ij}|_V \, \Delta\mathbf{V}.$$

The matrix is expressed as

$$|A_{ij}|_V = \frac{1}{c^2} \left[\begin{array}{ccc} C_1 & C_{234} & C_5 \end{array} \right],$$

being

$$\begin{aligned}
C_1 &= \left[\begin{array}{c} |\lambda_1| \, c^2 \\ |\lambda_1| \, c^2 \mathbf{v} \\ |\lambda_1| \, \dfrac{c^2 q^2}{2} \end{array} \right] \\
C_{234} &= \left[\begin{array}{c} \rho c \dfrac{1}{2} \left(|\lambda_2| - |\lambda_3| \right) \mathbf{n}^T \\ \rho c \dfrac{1}{2} \left(|\lambda_2| - |\lambda_3| \right) \mathbf{v} \otimes \mathbf{n} + \rho c^2 \dfrac{1}{2} \left(|\lambda_2| + |\lambda_3| \right) \mathbf{n} \otimes \mathbf{n} - \rho c^2 \, |\lambda_1| \, N \\ \left[\rho c H \dfrac{1}{2} \left(|\lambda_2| - |\lambda_3| \right) + \rho c^2 v_n \dfrac{1}{2} \left(|\lambda_2| + |\lambda_3| \right) \right] \mathbf{n}^T - \rho c^2 \, |\lambda_1| \, \mathbf{v}^T N \end{array} \right] \\
C_5 &= \left[\begin{array}{c} \dfrac{|\lambda_3| + |\lambda_2| - 2\,|\lambda_1|}{2} \\ \dfrac{|\lambda_3| + |\lambda_2| - 2\,|\lambda_1|}{2} \mathbf{v} + c \dfrac{1}{2} \left(|\lambda_2| - |\lambda_3| \right) \mathbf{n} \\ \dfrac{1}{2} \left(|\lambda_2| + |\lambda_3| \right) H - |\lambda_1| \, \dfrac{q^2}{2} + c v_n \dfrac{1}{2} \left(|\lambda_2| - |\lambda_3| \right) \end{array} \right]
\end{aligned}$$

where

$$N = \begin{bmatrix} -\left(1-n_x^2\right) & n_x n_y & n_x n_z \\ n_x n_y & -\left(1-n_y^2\right) & n_y n_z \\ n_x n_z & n_y n_z & -\left(1-n_z^2\right) \end{bmatrix},$$

the absolute eigenvalues are

$$\begin{aligned} |\lambda_1| &= |v_n| \\ |\lambda_2| &= |v_n + c| \\ |\lambda_3| &= |v_n - c| \end{aligned}$$

and the product $\mathbf{a} \otimes \mathbf{b} = a_i b_j$. All the variables in the matrix are considered as mean variables between the two nodes of the edge. The difference of the primitive variables is $\Delta\mathbf{V} = \mathbf{V}_j - \mathbf{V}_i$. The proposed linearised fluxes are

$$|A_{ij}|_V \, M^{-1} \Delta\mathbf{u}.$$

Thus, just the flow variables that affect the artificial viscosity matrix are linearised, but not the matrix itself. With this assumption, the contribution of the edge ij to the linearised artificial viscosity is, according to Eq. (2.10)

$$\mathbf{LF}_{ij}^{av} = |A_{ij}|_V \, M^{-1} \left[S_{ij} \left(\mathbf{u}_i - \mathbf{u}_j\right) - \frac{1}{2}\left(1-\kappa\right)\left(1-S_{ij}\right)\left[L_i\left(\mathbf{u}\right) - L_j\left(\mathbf{u}\right)\right] \right] \frac{\sigma_{ij}}{2}$$

where we have used the formulation with the pseudo-Laplacian operator $\mathbf{L}_i$. The switch S_{ij} has not been linearised, neither.

First, we derive the contribution of an edge ij to the second differences:

$$\begin{bmatrix} \mathbf{LF}_i^{av} \\ \mathbf{LF}_j^{av} \end{bmatrix}_{2nd} = \frac{\sigma_{ij}}{2} S_{ij} \begin{bmatrix} -|A_{ij}|_V & |A_{ij}|_V \\ |A_{ij}|_V & -|A_{ij}|_V \end{bmatrix} M^{-1} \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_j \end{bmatrix}, \tag{B.2}$$

which transposed turns into the edge contribution to the adjoint second differences:

$$\begin{bmatrix} \mathbf{LF}_i^{av} \\ \mathbf{LF}_j^{av} \end{bmatrix}_{2nd} = \frac{\sigma_{ij}}{2} S_{ij} \left(M^{-1}\right)^T \begin{bmatrix} -|A_{ij}|_V^T & |A_{ij}|_V^T \\ |A_{ij}|_V^T & -|A_{ij}|_V^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_i \\ \boldsymbol{\psi}_j \end{bmatrix}$$

The contribution for each of the nodes that conforms the edge is:

$$\mathbf{LF}^{av}_{i_{2nd}} = -S_{ij}\left(M_i^{-1}\right)^T |A_{ij}|_V^T \left(\boldsymbol{\psi}_i - \boldsymbol{\psi}_j\right)\frac{\sigma_{ij}}{2}$$
$$\mathbf{LF}^{av}_{j_{2nd}} = S_{ij}\left(M_j^{-1}\right)^T |A_{ij}|_V^T \left(\boldsymbol{\psi}_i - \boldsymbol{\psi}_j\right)\frac{\sigma_{ij}}{2}$$

The fourth differences terms have a structure that is analogous to that of the second differences of Eq. (B.2), but the pseudo-Laplacian of the variables is used instead of the variables themselves.

$$\begin{bmatrix} \mathbf{LF}^{av}_i \\ \mathbf{LF}^{av}_j \end{bmatrix}_{4th} = -\frac{\sigma_{ij}}{4}(1-\kappa)(1-S_{ij}) \begin{bmatrix} -|A_{ij}|_V & |A_{ij}|_V \\ |A_{ij}|_V & -|A_{ij}|_V \end{bmatrix} M^{-1} \begin{bmatrix} \mathbf{pL}_i \\ \mathbf{pL}_j \end{bmatrix} \tag{B.3}$$

The contribution of an edge ij to the pseudo-Laplacian of the linearised primitive variables is

$$\mathbf{pL}_{ij} = \frac{1}{\#ed_j}\mathbf{u}_j - \frac{1}{\#ed_i}\mathbf{u}_i \tag{B.4}$$

which yields a contribution to the nodes i and j given by:

$$\begin{bmatrix} \mathbf{pL}_i \\ \mathbf{pL}_j \end{bmatrix} = \begin{bmatrix} \frac{1}{\#ed} \end{bmatrix} \begin{bmatrix} -I & I \\ I & -I \end{bmatrix} \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_j \end{bmatrix}$$

where $\#ed$ is the number of edges that reach a node. The complete operator is a sum over edges of the edge contributions of Eq. (B.4):

$$[pL] = \sum_{kl=1}^{Nedges} \mathbf{pL}_{kl} = \begin{bmatrix} \frac{1}{\#ed} \end{bmatrix} \left\{ \sum_{kl=1}^{Nedges} \begin{bmatrix} -I & I \\ I & -I \end{bmatrix} \right\}$$

where $Nedges$ is the number of edges of the grid. The transposed of this expression yields the adjoint pseudo-Laplacian operator:

$$[pL]^T = \left\{ \sum_{kl=1}^{Nedges} \begin{bmatrix} -I & I \\ I & -I \end{bmatrix} \right\} \begin{bmatrix} \frac{1}{\#ed} \end{bmatrix}$$

The contribution of the edge ij to the adjoint fourth differences yields

$$\begin{bmatrix} \mathbf{AF}^{av}_i \\ \mathbf{AF}^{av}_j \end{bmatrix}_{4th} = -\frac{\sigma_{ij}}{4}(1-\kappa)(1-S_{ij})\left(M^{-1}\right)^T [pL]^T \begin{bmatrix} -|A_{ij}|_V^T & |A_{ij}|_V^T \\ |A_{ij}|_V^T & -|A_{ij}|_V^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_i \\ \boldsymbol{\psi}_j \end{bmatrix}$$

If we sum over all edges and rearrange the result we finally obtain the fourth differences contribution to the adjoint numerical diffusion:

$$\mathbf{AF}_{4th}^{av} = \left(M^{-1}\right)^T \left[[pL]^T \left\{ \sum_{ij=1}^{Nedges} k\frac{\sigma_{ij}}{2}\left(1-\Psi_{ij}\right) \begin{bmatrix} -\left|A_{ij}\right|_V^T & \left|A_{ij}\right|_V^T \\ \left|A_{ij}\right|_V^T & -\left|A_{ij}\right|_V^T \end{bmatrix} \right\} \right] \begin{bmatrix} \boldsymbol{\psi}_i \\ \boldsymbol{\psi}_j \end{bmatrix}$$

The transposition affects the order of evaluation of the terms. First, the second differences contribution to the numerical diffusion must be computed, and then, the pseudo-Laplacian is applied over them, the opposite to what happens in the linear case.

The contribution of an edge ij to the second differences adjoint numerical diffusion is provided by the product $\left|A_{ij}\right|_V^T \Delta\boldsymbol{\psi}$, that results in

$$\begin{aligned}
f_1 &= \left[\left|\lambda_1\right| c^2 \left(\Delta\widetilde{\psi}_1 + \mathbf{v}\cdot\Delta\widetilde{\boldsymbol{\psi}}_{234}\right)\right]\frac{1}{c^2} \\
f_{234} &= \left\{\left[\rho c\frac{1}{2}\left(\left|\lambda_2\right|-\left|\lambda_3\right|\right)\left(\Delta\widetilde{\psi}_1 + \mathbf{v}\cdot\Delta\widetilde{\boldsymbol{\psi}}_{234} + \frac{c^2}{\gamma-1}\Delta\psi_5\right) + \rho c^2\frac{\left|\lambda_3\right|+\left|\lambda_2\right|-2\left|\lambda_1\right|}{2}\mathbf{n}\cdot\Delta\widetilde{\boldsymbol{\psi}}_{234}\right]\mathbf{n}+\right. \\
&+ \left.\rho c^2\left|\lambda_1\right|\Delta\widetilde{\boldsymbol{\psi}}_{234}\right\}\frac{1}{c^2} \\
f_5 &= \left[\frac{\left|\lambda_3\right|+\left|\lambda_2\right|-2\left|\lambda_1\right|}{2}\left(\Delta\widetilde{\psi}_1 + \mathbf{v}\cdot\Delta\widetilde{\boldsymbol{\psi}}_{234}\right) + c\frac{1}{2}\left(\left|\lambda_2\right|-\left|\lambda_3\right|\right)\mathbf{n}\cdot\Delta\widetilde{\boldsymbol{\psi}}_{234}+\right. \\
&+ \left.\frac{1}{2}\left(\left|\lambda_2\right|+\left|\lambda_3\right|\right)\frac{c^2}{\gamma-1}\Delta\psi_5\frac{1}{c^2}\right]
\end{aligned}$$

B.3. Adjoint viscous fluxes

The linearised viscous fluxes can be represented as a product of two matrices, one accounting for the evaluation of the stress tensor and the other for the evaluation of the viscous terms themselves. To show this, let us start from the $\mathbf{F}_v\cdot\mathbf{n}$ product of Eq. (2.6). This vector can be split into a sum of three vectors, each involving the derivatives in x, y and z directions, respectively. The particularisation of this product for an edge ij may then be written as

$$\mathbf{F}_{ij}^v\cdot\mathbf{n}_{ij} = B_{ij}^{(x)}\left.\frac{\partial\mathbf{V}}{\partial x}\right|_{ij} + B_{ij}^{(y)}\left.\frac{\partial\mathbf{V}}{\partial y}\right|_{ij} + B_{ij}^{(z)}\left.\frac{\partial\mathbf{V}}{\partial z}\right|_{ij} \tag{B.5}$$

with

$$B_{ij}^{(x)} = \mu \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{4}{3}n_x & n_y & n_z & 0 \\ 0 & -\frac{2}{3}n_y & n_x & 0 & 0 \\ 0 & -\frac{2}{3}n_z & 0 & n_x & 0 \\ -\frac{1}{Pr}\frac{\gamma}{\gamma-1}\frac{p}{\rho^2}n_x & \frac{4}{3}un_x - \frac{2}{3}vn_y - \frac{2}{3}wn_z & un_y + vn_x & un_z + wn_x & \frac{1}{Pr}\frac{\gamma}{\gamma-1}\frac{1}{\rho}n_x \end{bmatrix}$$

$$B_{ij}^{(y)} = \mu \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & n_y & -\frac{2}{3}n_x & 0 & 0 \\ 0 & n_x & \frac{4}{3}n_y & n_z & 0 \\ 0 & 0 & -\frac{2}{3}n_z & n_y & 0 \\ -\frac{1}{Pr}\frac{\gamma}{\gamma-1}\frac{p}{\rho^2}n_y & un_y + vn_x & -\frac{2}{3}un_x + \frac{4}{3}vn_y - \frac{2}{3}wn_z & vn_z + wn_y & \frac{1}{Pr}\frac{\gamma}{\gamma-1}\frac{1}{\rho}n_y \end{bmatrix}$$

$$B_{ij}^{(z)} = \mu \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & n_z & 0 & -\frac{2}{3}n_x & 0 \\ 0 & 0 & n_z & -\frac{2}{3}n_y & 0 \\ 0 & n_x & n_y & \frac{4}{3}n_z & 0 \\ -\frac{1}{Pr}\frac{\gamma}{\gamma-1}\frac{p}{\rho^2}n_z & un_z + wn_x & vn_z + wn_y & -\frac{2}{3}un_x - \frac{2}{3}vn_y + \frac{4}{3}wn_z & \frac{1}{Pr}\frac{\gamma}{\gamma-1}\frac{1}{\rho}n_z \end{bmatrix}$$

The value of each variable in the edge ij is computed as a mean between the nodes i and j that conform it, and the gradients of the variables are evaluated with the formula of Eq. (2.15). Linearising Eq. (B.5) and using the Eq. (2.7), that yields the contribution of each edge to the inviscid and viscous fluxes, we obtain the edge contribution of an edge ij to the linearised viscous fluxes:

$$\begin{aligned} \mathbf{LF}_{ij}^{v} &= B_{ij}^{(x)} M^{-1} \left.\frac{\partial \mathbf{u}}{\partial x}\right|_{ij} + B_{ij}^{(y)} M^{-1} \left.\frac{\partial \mathbf{u}}{\partial y}\right|_{ij} + B_{ij}^{(z)} M^{-1} \left.\frac{\partial \mathbf{u}}{\partial z}\right|_{ij} + \\ &+ \frac{1}{2}\left[\frac{\partial}{\partial \mathbf{V}}\left(B_{ij}^{(x)} \left.\frac{\partial \mathbf{V}}{\partial x}\right|_{ij} + B_{ij}^{(y)} \left.\frac{\partial \mathbf{V}}{\partial y}\right|_{ij} + B_{ij}^{(z)} \left.\frac{\partial \mathbf{V}}{\partial z}\right|_{ij}\right)\right] M^{-1} \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_j \end{bmatrix} \end{aligned}$$

The gradient of a variable is expressed as a product of a sparse matrix with the information of the nodes contained in every edge and the vector of discretized variables,

$$\frac{\partial \mathbf{V}}{\partial x} = \left\{ \sum_{ij=1}^{Nedges} D_{ij}^{(x)} \right\} \mathbf{V}$$

where $D_{ij}^{(x)}$ stands for the contribution of the edge ij to the x component of the gradient,

$$D_{ij}^{(x)} = \frac{1}{2}\begin{bmatrix} \frac{I}{V_i} & -\frac{I}{V_i} \\ \frac{I}{V_j} & -\frac{I}{V_j} \end{bmatrix} \sigma_x n_x$$

The expression is analogous for the $D^{(y)}$ and $D^{(z)}$ operators. The transposed of this antisymmetric matrix is

$$\left(D_{ij}^{(x)}\right)^T = \frac{1}{2}\begin{bmatrix} \frac{I}{V_i} & \frac{I}{V_j} \\ -\frac{I}{V_i} & -\frac{I}{V_j} \end{bmatrix} \sigma_x n_x,$$

yielding an expression for the x component of the adjoint gradient:

$$\left(D^{(x)}\right)^T = \sum_{ij=1}^{Nedges} \left(D_{ij}^{(x)}\right)^T$$

The gradient in the edge is provided by Eq. (2.15), that can be expressed as

$$\begin{aligned}
\left.\frac{\partial \mathbf{u}}{\partial x}\right|_{ij} &= \left\{\frac{1}{2}\begin{bmatrix} I_i & I_j \end{bmatrix}\left[D^{(x)} - \Psi \cdot \Delta x\right] + \frac{\Delta x}{\Delta l^2}\begin{bmatrix} I_i & -I_j \end{bmatrix}\right\}\begin{bmatrix} \mathbf{u} \end{bmatrix} \\
\left.\frac{\partial \mathbf{u}}{\partial y}\right|_{ij} &= \left\{\frac{1}{2}\begin{bmatrix} I_i & I_j \end{bmatrix}\left[D^{(y)} - \Psi \cdot \Delta y\right] + \frac{\Delta y}{\Delta l^2}\begin{bmatrix} I_i & -I_j \end{bmatrix}\right\}\begin{bmatrix} \mathbf{u} \end{bmatrix} \\
\left.\frac{\partial \mathbf{u}}{\partial z}\right|_{ij} &= \left\{\frac{1}{2}\begin{bmatrix} I_i & I_j \end{bmatrix}\left[D^{(z)} - \Psi \cdot \Delta z\right] + \frac{\Delta z}{\Delta l^2}\begin{bmatrix} I_i & -I_j \end{bmatrix}\right\}\begin{bmatrix} \mathbf{u} \end{bmatrix}
\end{aligned}$$

where

$$\Psi = \frac{1}{\Delta l^2}\left[D^{(x)}\Delta x + D^{(y)}\Delta y + D^{(z)}\Delta z\right]$$

The complete formula of the contribution to the linearised viscous fluxes of nodes i and j is

$$\begin{aligned}
\begin{bmatrix} \mathbf{LF}_i^v \\ \mathbf{LF}_j^v \end{bmatrix} &= \sigma_{ij}\left\{\frac{1}{2}\Phi_{ij}\begin{bmatrix} I & I \\ -I & -I \end{bmatrix} + \right. \\
&+ \frac{1}{2}B_{ij}^{(x)}\begin{bmatrix} I & I \\ -I & -I \end{bmatrix}\left[D^{(x)} - \Psi \cdot \Delta x\right] + \\
&+ \left.\frac{1}{2}B_{ij}^{(y)}\begin{bmatrix} I & I \\ -I & -I \end{bmatrix}\left[D^{(y)} - \Psi \cdot \Delta y\right] + \right.
\end{aligned}$$

$$
\begin{aligned}
&+ \quad \frac{1}{2} B_{ij}^{(z)} \begin{bmatrix} I & I \\ -I & -I \end{bmatrix} \left[D^{(z)} - \Psi \cdot \Delta z \right] + \\
&+ \quad \left[\frac{\Delta x}{\Delta l^2} B_{ij}^{(x)} + \frac{\Delta y}{\Delta l^2} B_{ij}^{(y)} + \frac{\Delta z}{\Delta l^2} B_{ij}^{(z)} \right] \begin{bmatrix} I & -I \\ -I & I \end{bmatrix} \Bigg] M^{-1} \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_j \end{bmatrix}
\end{aligned}
$$

where

$$
\Phi_{ij} = \frac{\partial}{\partial \mathbf{V}} \left(B_{ij}^{(x)} \left. \frac{\partial \mathbf{V}}{\partial x} \right|_{ij} + B_{ij}^{(y)} \left. \frac{\partial \mathbf{V}}{\partial y} \right|_{ij} + B_{ij}^{(z)} \left. \frac{\partial \mathbf{V}}{\partial z} \right|_{ij} \right) =
$$

$$
= \begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
-\frac{\mu}{Pr} \frac{\gamma}{\gamma - 1} \frac{1}{\rho^2} \left(\nabla p \cdot \mathbf{n} - \frac{2p}{\rho} \nabla \rho \cdot \mathbf{n} \right) & \tau_{xn} & \tau_{yn} & \tau_{zn} & -\frac{\mu}{Pr} \frac{\gamma}{\gamma - 1} \frac{1}{\rho^2} \nabla \rho \cdot \mathbf{n}
\end{bmatrix} +
$$

$$
+ \begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
\frac{\partial \mu}{\partial \rho} \tau_{xn} & 0 & 0 & 0 & \frac{\partial \mu}{\partial p} \tau_{xn} \\
\frac{\partial \mu}{\partial \rho} \tau_{yn} & 0 & 0 & 0 & \frac{\partial \mu}{\partial p} \tau_{yn} \\
\frac{\partial \mu}{\partial \rho} \tau_{zn} & 0 & 0 & 0 & \frac{\partial \mu}{\partial p} \tau_{zn} \\
\frac{\partial \mu}{\partial \rho} \mathbf{F}_v \cdot \mathbf{n}|_5 & 0 & 0 & 0 & \frac{\partial \mu}{\partial p} \mathbf{F}_v \cdot \mathbf{n}|_5
\end{bmatrix},
$$

being

$$
\begin{aligned}
\frac{\partial \mu}{\partial \rho} &= -\frac{\mu}{\rho} \left(\frac{3}{2} - \frac{T}{T + 110.4} \right) \\
\frac{\partial \mu}{\partial p} &= \frac{\mu}{p} \left(\frac{3}{2} - \frac{T}{T + 110.4} \right)
\end{aligned}
$$

the derivatives of the Sutherland law.

If we now transpose the linear flux, we obtain the contribution of the edge ij to the adjoint viscous flux of nodes i and j:

$$
\begin{aligned}
\begin{bmatrix} \mathbf{AF}_i^v \\ \mathbf{AF}_j^v \end{bmatrix} &= \left(M^{-1}\right)^T \sigma_{ij} \left\{ \frac{1}{2}\Phi_{ij}^T \begin{bmatrix} I & -I \\ I & -I \end{bmatrix} + \frac{1}{2}\left\{ \left[\left(D^{(x)}\right)^T - \Psi^T \cdot \Delta x \right] \left(B_{ij}^{(x)}\right)^T + \right.\right. \\
&+ \left[\left(D^{(y)}\right)^T - \Psi^T \cdot \Delta y \right] \left(B_{ij}^{(y)}\right)^T + \left[\left(D^{(z)}\right)^T - \Psi^T \cdot \Delta z \right] \left(B_{ij}^{(z)}\right)^T \Bigg\} \begin{bmatrix} I & -I \\ I & -I \end{bmatrix} + \\
&+ \left.\left. \begin{bmatrix} I & -I \\ -I & I \end{bmatrix} \left[\frac{\Delta x}{\Delta l^2} \left(B_{ij}^{(x)}\right)^T + \frac{\Delta y}{\Delta l^2} \left(B_{ij}^{(y)}\right)^T + \frac{\Delta z}{\Delta l^2} \left(B_{ij}^{(z)}\right)^T \right] \right\} \begin{bmatrix} \boldsymbol{\psi}_i \\ \boldsymbol{\psi}_j \end{bmatrix}
\end{aligned}
$$

The expressions of the matrix times vector products are

$$
\left(B_{ij}^{(x)}\right)^T \Delta\boldsymbol{\psi} = \begin{bmatrix} -\dfrac{\mu}{Pr}\dfrac{\gamma}{\gamma-1}\dfrac{p}{\rho^2}\Delta\psi_5 n_x \\ \alpha_{xx} \\ \alpha_{xy} \\ \alpha_{xz} \\ \dfrac{\mu}{Pr}\dfrac{\gamma}{\gamma-1}\dfrac{1}{\rho}\Delta\psi_5 n_x \end{bmatrix}
$$

$$
\left(B_{ij}^{(y)}\right)^T \Delta\boldsymbol{\psi} = \begin{bmatrix} -\dfrac{\mu}{Pr}\dfrac{\gamma}{\gamma-1}\dfrac{p}{\rho^2}\Delta\psi_5 n_y \\ \alpha_{xy} \\ \alpha_{yy} \\ \alpha_{yz} \\ \dfrac{\mu}{Pr}\dfrac{\gamma}{\gamma-1}\dfrac{1}{\rho}\Delta\psi_5 n_y \end{bmatrix}
$$

$$
\left(B_{ij}^{(z)}\right)^T \Delta\boldsymbol{\psi} = \begin{bmatrix} -\dfrac{\mu}{Pr}\dfrac{\gamma}{\gamma-1}\dfrac{p}{\rho^2}\Delta\psi_5 n_z \\ \alpha_{xz} \\ \alpha_{yz} \\ \alpha_{zz} \\ \dfrac{\mu}{Pr}\dfrac{\gamma}{\gamma-1}\dfrac{1}{\rho}\Delta\psi_5 n_z \end{bmatrix}
$$

where

$$
\begin{aligned}
\alpha_{xx} &= \frac{2\mu}{3}\left(2\Delta\widetilde{\psi}_2 n_x - \Delta\widetilde{\psi}_3 n_y - \Delta\widetilde{\psi}_4 n_z\right) \\
\alpha_{yy} &= \frac{2\mu}{3}\left(2\Delta\widetilde{\psi}_3 n_y - \Delta\widetilde{\psi}_2 n_x - \Delta\widetilde{\psi}_4 n_z\right)
\end{aligned}
$$

$$
\begin{aligned}
\alpha_{zz} &= \frac{2\mu}{3}\left(2\Delta\widetilde{\psi}_4 n_z - \Delta\widetilde{\psi}_2 n_x - \Delta\widetilde{\psi}_3 n_y\right) \\
\alpha_{xy} &= \mu\left(\Delta\widetilde{\psi}_2 n_y + \Delta\widetilde{\psi}_3 n_x\right) \\
\alpha_{xz} &= \mu\left(\Delta\widetilde{\psi}_2 n_z + \Delta\widetilde{\psi}_4 n_x\right) \\
\alpha_{yz} &= \mu\left(\Delta\widetilde{\psi}_3 n_z + \Delta\widetilde{\psi}_4 n_y\right)
\end{aligned}
$$

could represent a pseudo-stress tensor of the adjoint variables. The other term is:

$$
\Phi_{ij}^T \Delta\boldsymbol{\psi} = \begin{bmatrix} -\dfrac{\mu}{Pr}\dfrac{\gamma}{\gamma-1}\dfrac{1}{\rho^2}\left(\nabla p\cdot\mathbf{n} - \dfrac{2p}{\rho}\nabla\rho\cdot\mathbf{n}\right) \\ \tau_{xn} \\ \tau_{yn} \\ \tau_{zn} \\ -\dfrac{\mu}{Pr}\dfrac{\gamma}{\gamma-1}\dfrac{1}{\rho^2}\nabla\rho\cdot\mathbf{n} \end{bmatrix} \Delta\psi_5 + \begin{bmatrix} \dfrac{\partial\mu}{\partial\rho}\Xi \\ 0 \\ 0 \\ 0 \\ \dfrac{\partial\mu}{\partial p}\Xi \end{bmatrix}
$$

with $\Xi = \Delta\widetilde{\psi}_2\tau_{xn} + \Delta\widetilde{\psi}_3\tau_{yn} + \Delta\widetilde{\psi}_4\tau_{zn} + \Delta\psi_5\dfrac{\mu}{Pr}\dfrac{\gamma}{\gamma-1}\left(\dfrac{1}{\rho}\nabla p - \dfrac{p}{\rho^2}\nabla\rho\right)\cdot\mathbf{n}$

B.4. Adjoint Inlet and Outlet Boundary Conditions

The inlet and outlet boundary conditions which are first linearised, and then transposed, correspond to the one dimensional non reflecting boundary conditions formulation. Two relations are useful for the obtention of the boundary conditions: the linearised state equation and the linearised Riemann invariants. The linearisation of the state equation $p_0 = \rho_0 R T_0$ yields

$$
\frac{dp}{p_0} = \frac{d\rho}{\rho_0} + \frac{dT}{T_0} \tag{B.6}
$$

and the first and second Riemann invariants

$$
R_\pm = v_n \pm \frac{2c}{\gamma-1}
$$

yield

$$
dR_\pm = dv_n \pm \frac{c_0}{\gamma-1}\left(\frac{dp}{p_0} - \frac{d\rho}{\rho_0}\right) \tag{B.7}
$$

B.4.1. Subsonic inlet

The flow conditions for a subsonic inlet are determined by imposing the total pressure, total temperature and inlet flow angle (radial and tangential). The additional condition is that the Riemann invariant R_- is extrapolated from inside the computational domain. The Riemann invariant yields an equation that relates v_n and T:

$$v_n^2 = R_-^2 + \frac{4}{(\gamma-1)^2}\gamma R_g T + \frac{4}{\gamma-1}\sqrt{\gamma R_g T} \tag{B.8}$$

The other relation between v_n and T is provided by the expression of the total temperature:

$$T_t = T + \frac{q^2}{2C_p} = T + \frac{1}{2C_p}v_n^2\left(1+\tan^2\alpha\right) \tag{B.9}$$

since

$$q^2 = v_n^2 + v_t^2$$

and

$$\tan\alpha = \frac{|v_t|}{|v_n|}$$

being $\cos\alpha = \cos\beta_s\cos\beta_r\cdot n_x + \sin\beta_s\cdot n_y + \cos\beta_s\sin\beta_r\cdot n_z$ the angle between the velocity vector and the inlet area normal (β_s and β_r are the swirl and radial angles at the inlet).

Combining Eqs. (B.8) and (B.9), we obtain the following expression for the temperature:

$$AT + BT^{1/2} + C = 0$$

where

$$\begin{aligned} A &= 1 + \frac{2}{\gamma-1}\frac{1}{\cos^2\alpha} \\ B &= \frac{2}{\cos^2\alpha}\frac{1}{\sqrt{\gamma R_g}}R_- \\ C &= \frac{1}{2C_p\cos^2\alpha}R_-^2 - T_t \end{aligned}$$

The linearised equation results in

$$\frac{dT^*}{T_0} = -\frac{(\gamma-1)\,M_{n0}}{M_{n0}+\cos^2\alpha}\frac{1}{c_0}dR_- = -(\gamma-1)\,\lambda\frac{1}{c_0}dR_-,$$

being

$$\lambda = \frac{M_{n0}}{M_{n0} + \cos^2 \alpha}$$

The equation that relates p_t and T_t, assuming that these two values remain constant, provides us the desired dp value:

$$dp^* = p_0 \frac{\gamma}{\gamma - 1} \frac{dT^*}{T_0}$$

and the new $d\rho$ is obtained with the linearised state equation (Eq. (B.6)):

$$d\rho^* = \rho_0 \frac{1}{\gamma - 1} \frac{dT^*}{T_0}$$

Finally, Eq. (B.7) provides the new normal velocity:

$$dv_n^* = dR_- + \frac{c_0}{\gamma - 1} \frac{dT^*}{T_0}$$

To obtain the three components of the velocity we have to multiply the dv_n^* by a factor depending on the angle of the velocity vector and the normal, thus $d\mathbf{v} = [\alpha_u \ \alpha_v \ \alpha_w]\, dv_n^*$, where

$$\begin{aligned}
\alpha_u &= \frac{\cos\beta_s \cos\beta_r}{\cos\alpha_n} \\
\alpha_v &= \frac{\sin\beta_s \cos\theta + \cos\beta_s \sin\beta_r \sin\theta}{\cos\alpha_n} \\
\alpha_w &= \frac{-\sin\beta_s \sin\theta + \cos\beta_s \sin\beta_r \cos\theta}{\cos\alpha_n}
\end{aligned}$$

and

$$\begin{aligned}
\cos\alpha_n &= \cos\beta_r \cos\beta_s \cdot n_x + \sin\beta_s \cdot n_y + \sin\beta_r \cos\beta_s \cdot n_z \\
\theta &= \arctan\frac{y}{z}
\end{aligned}$$

The boundary conditions can be expressed as:

$$\begin{bmatrix} d\rho \\ du \\ dv \\ dw \\ dp \end{bmatrix}^* = \begin{bmatrix} \phi_1 & 0 & 0 & 0 & 0 \\ 0 & \phi_2 & 0 & 0 & 0 \\ 0 & 0 & \phi_3 & 0 & 0 \\ 0 & 0 & 0 & \phi_4 & 0 \\ 0 & 0 & 0 & 0 & \phi_5 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \chi_1 & 0 & 0 & 0 & 0 \\ 0 & \chi_2 & 0 & 0 & 0 \\ 0 & 0 & \chi_3 & 0 & 0 \\ 0 & 0 & 0 & \chi_4 & 0 \\ 0 & 0 & 0 & 0 & \chi_5 \end{bmatrix} \begin{bmatrix} d\rho \\ du \\ dv \\ dw \\ dp \end{bmatrix}$$

being

$$\begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \end{bmatrix} = \begin{bmatrix} -\dfrac{\lambda\rho_0}{c_0} \\ (1-\lambda)\,\alpha_u \\ (1-\lambda)\,\alpha_v \\ (1-\lambda)\,\alpha_w \\ -\dfrac{\lambda\gamma p_0}{c_0} \end{bmatrix},$$

and

$$\begin{bmatrix} \chi_1 \\ \chi_2 \\ \chi_3 \\ \chi_4 \\ \chi_5 \end{bmatrix} = \begin{bmatrix} \dfrac{c_0}{\gamma-1}\dfrac{1}{\rho_0} \\ n_x \\ n_y \\ n_z \\ -\dfrac{c_0}{\gamma-1}\dfrac{1}{p_0} \end{bmatrix}$$

The conditions must be written in conservative variables, since it this the way they are used in conjunction with the above described fluxes. Thus,

$$\begin{bmatrix} d\rho \\ d(\rho u) \\ d(\rho v) \\ d(\rho w) \\ d(\rho E) \end{bmatrix}^* = M[\phi] \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} [\chi]\, M^{-1} \begin{bmatrix} d\rho \\ d(\rho u) \\ d(\rho v) \\ d(\rho w) \\ d(\rho E) \end{bmatrix}$$

The linearised transposed boundary conditions will then be

$$\begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \\ \psi_5 \end{bmatrix}^* = \left(M^{-1}\right)^T [\chi] \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} [\phi]\, M^T \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \\ \psi_5 \end{bmatrix},$$

that developed gives

$$\begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \\ \psi_5 \end{bmatrix}^* = \begin{bmatrix} \chi_{AD1} \\ \chi_{AD2} \\ \chi_{AD3} \\ \chi_{AD4} \\ \chi_{AD5} \end{bmatrix} R_{AD}$$

being

$$R_{AD} = \left(-\frac{\lambda\rho_0}{c_0} \left(\widetilde{\psi}_1 + \mathbf{v} \cdot \widetilde{\boldsymbol{\psi}}_{234} \right) + \rho \left(1 - \lambda \right) \boldsymbol{\alpha} \cdot \widetilde{\boldsymbol{\psi}}_{234} - \frac{\gamma}{\gamma - 1} \frac{\lambda p_0}{c_0} \psi_5 \right),$$

$$\begin{bmatrix} \chi_{AD1} \\ \chi_{AD2} \\ \chi_{AD3} \\ \chi_{AD4} \\ \chi_{AD5} \end{bmatrix} = \begin{bmatrix} \frac{c_0}{\gamma - 1} \frac{1}{\rho_0} \left(1 - (\gamma - 1) \frac{v_{n0}}{c_0} - \frac{\gamma - 1}{2} \frac{\rho_0 q^2}{p_0} \right) \\ \frac{n_x}{\rho_0} + \frac{c_0 u_0}{p_0} \\ \frac{n_y}{\rho_0} + \frac{c_0 v_0}{p_0} \\ \frac{n_z}{\rho_0} + \frac{c_0 w_0}{p_0} \\ -\frac{c_0}{p_0} \end{bmatrix}$$

and $\boldsymbol{\alpha} = [\alpha_u \ \alpha_v \ \alpha_w]$.

B.4.2. Subsonic outlet

The first necessary relation to calculate the new outlet state is obtained by linearising the third Riemann invariant. Assuming that the steady static pressure at the outlet remains fixed, the new linearised density is

$$d\rho^* = d\rho - \frac{\rho_0}{\gamma} \frac{dp}{p_0} = d\rho - \frac{dp}{c_0^2}$$

The new linearised velocity is $d\mathbf{v}^* = d\mathbf{v} + (dv_n^* - dv_n)\,\mathbf{n}$. The normal velocity is obtained with the Riemann invariant R_+ of Eq. (B.7), hence providing a difference

$$dv_n^* - dv_n = \frac{c_0}{\gamma} \frac{dp}{p_0} = \frac{dp}{\rho_0 c_0}$$

And last

$$dp^* = 0$$

All these formula can also be expressed as,

$$\begin{bmatrix} d\rho \\ d(\rho u) \\ d(\rho v) \\ d(\rho w) \\ d(\rho E) \end{bmatrix}^* = M \begin{bmatrix} 1 & 0 & 0 & 0 & -\frac{1}{c_0^2} \\ 0 & 1 & 0 & 0 & \frac{n_x}{\rho_0 c_0} \\ 0 & 0 & 1 & 0 & \frac{n_y}{\rho_0 c_0} \\ 0 & 0 & 0 & 1 & \frac{n_z}{\rho_0 c_0} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} M^{-1} \begin{bmatrix} d\rho \\ d(\rho u) \\ d(\rho v) \\ d(\rho w) \\ d(\rho E) \end{bmatrix},$$

which transposed yields

$$\begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \\ \psi_5 \end{bmatrix}^* = \left(M^{-1}\right)^T \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -\frac{1}{c_0^2} & \frac{n_x}{\rho_0 c_0} & \frac{n_y}{\rho_0 c_0} & \frac{n_z}{\rho_0 c_0} & 0 \end{bmatrix} M^T \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \\ \psi_5 \end{bmatrix}$$

Operating with the matrices we finally obtain

$$\begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \\ \psi_5 \end{bmatrix}^* = \begin{bmatrix} \widetilde{\psi}_1 - \frac{\gamma-1}{2}\frac{q^2}{c_0^2}\left[\widetilde{\psi}_1 + \widetilde{\boldsymbol{\psi}}_{234}\cdot(\mathbf{v}-c_0\mathbf{n})\right] \\ \widetilde{\psi}_2 + \frac{\gamma-1}{c_0^2}u\left[\widetilde{\psi}_1 + \widetilde{\boldsymbol{\psi}}_{234}\cdot(\mathbf{v}-c_0\mathbf{n})\right] \\ \widetilde{\psi}_3 + \frac{\gamma-1}{c_0^2}v\left[\widetilde{\psi}_1 + \widetilde{\boldsymbol{\psi}}_{234}\cdot(\mathbf{v}-c_0\mathbf{n})\right] \\ \widetilde{\psi}_4 + \frac{\gamma-1}{c_0^2}w\left[\widetilde{\psi}_1 + \widetilde{\boldsymbol{\psi}}_{234}\cdot(\mathbf{v}-c_0\mathbf{n})\right] \\ -\frac{\gamma-1}{c_0^2}\left[\widetilde{\psi}_1 + \widetilde{\boldsymbol{\psi}}_{234}\cdot(\mathbf{v}-c_0\mathbf{n})\right] \end{bmatrix}$$

B.5. Resolution of the adjoint Navier-Stokes equations

The expression of an adjoint iterative scheme starting from its linear counterpart has been obtained by Giles [33]. The starting point is the general discrete equation for the explicit integration of the system of equations, expressed, for a linear system of equations $L\mathbf{u} = \mathbf{f}$, as

$$\mathbf{u}^{n+1} = \mathbf{u}^n + R\left(\mathbf{f} - L\mathbf{u}^n\right) \tag{B.10}$$

where $\mathbf{u}^0 = 0$ and P is a preconditioning matrix. R is a matrix that represents an iterative process, such as Runge-Kutta or multigrid. When converged, $\mathbf{u}^{n+1} = \mathbf{u}^n$ and then $\mathbf{f} = L\mathbf{u}^n$

provides the solution to the steady problem. In the preceding expressions, L is the linearised matrix of the original non-linear system of equations, i.e.:

$$L = \left[\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_j}\right],$$

$\mathbf{u} = \partial \mathbf{U}/\partial \boldsymbol{\varphi}_k$ and $\mathbf{f}$ is the forcing term resulting from the perturbation of the system of equations with the geometric parameters,

$$\mathbf{f} = \frac{\partial \mathbf{R}_i}{\partial \boldsymbol{\varphi}_k}$$

To obtain the adjoint time marching procedure we start formulating the gradient of the objective function of Eq. (8.5),

$$I^N = \mathbf{g}^T \cdot \mathbf{u}^N \tag{B.11}$$

where $\mathbf{g} = \partial f/\partial \mathbf{U}_j$ is the derivative of the cost function with respect to the conservative variables. We have omitted the term $\partial f/\partial \boldsymbol{\varphi}_k$ in Eq. (B.11) for the sake of brevity. By introducing the Lagrange multipliers $\mathbf{w}$, the previous equation can be re-written as

$$\begin{aligned} I^N &= \mathbf{g}^T \cdot \mathbf{u}^N - \sum_{n=0}^{N-1} \left(\mathbf{w}^{n+1}\right)^T \cdot \left(\mathbf{u}^{n+1} - \mathbf{u}^n - R\left(\mathbf{f} - L\mathbf{u}^n\right)\right) \\ &= \left(\mathbf{g} - \mathbf{w}^N\right)^T \cdot \mathbf{u}^N - \sum_{n=0}^{N-1} \left[(\mathbf{u}^n)^T \cdot \left(-\mathbf{w}^{n+1} + \mathbf{w}^n + L^T R^T \mathbf{w}^{n+1}\right) + \mathbf{f}^T \cdot R^T \cdot \mathbf{w}^{n+1}\right] \end{aligned}$$

making use of the expression

$$\sum_{n=0}^{N-1} \left(\mathbf{w}^{n+1}\right)^T \cdot \left(\mathbf{u}^{n+1} - \mathbf{u}^n\right) = \left(\mathbf{w}^N\right)^T \cdot \mathbf{u}^N - \left(\mathbf{w}^0\right)^T \cdot \mathbf{u}^0 - \sum_{n=0}^{N-1} \left(\mathbf{u}^n\right)^T \cdot \left(\mathbf{w}^{n+1} - \mathbf{w}^n\right)$$

Then the cost function variation of Eq. (B.11) is

$$I^N = \sum_{n=0}^{N-1} \mathbf{f}^T \cdot R^T \cdot \mathbf{w}^{n+1}$$

when the adjoint equation is fulfilled, i.e., when

$$\mathbf{w}^n = \mathbf{w}^{n+1} - L^T R^T \mathbf{w}^{n+1}$$

subject to the final value $\mathbf{w}^N = \mathbf{g}$. If we introduce a change of variable

$$\boldsymbol{\psi}^n = \sum_{m=n}^{N-1} R^T \mathbf{w}^{m+1},$$

the functional $I^N = \mathbf{f}^T \cdot \boldsymbol{\psi}^0$ and the equation

$$\begin{aligned}
\boldsymbol{\psi}^n - \boldsymbol{\psi}^{n+1} &= R^T \mathbf{w}^{n+1} \\
&= R^T \left(\mathbf{g} - \sum_{m=n+1}^{N-1} \mathbf{w}^{m+1} - \mathbf{w}^m \right) \\
&= R^T \left(\mathbf{g} - \sum_{m=n+1}^{N-1} L^T R^T \mathbf{w}^{m+1} \right) \\
&= R^T \left(\mathbf{g} - L^T \boldsymbol{\psi}^{n+1} \right)
\end{aligned}$$

Thus, the final adjoint equation is

$$\boldsymbol{\psi}^n = \boldsymbol{\psi}^{n+1} + R^T \left(\mathbf{g} - L^T \boldsymbol{\psi}^{n+1} \right) \tag{B.12}$$

subject to the final condition $\boldsymbol{\psi}^N = 0$. This expression shows that the adjoint way of iterating a linear system like Eq. (B.10) is by using the adjoint matrix of R as an iterative scheme. When the adjoint problem is converged, $\boldsymbol{\psi}^n = \boldsymbol{\psi}^{n+1}$ and the adjoint steady solution $\mathbf{g} - L^T \boldsymbol{\psi}^{n+1} = 0$ is recovered. This general result concerning the equivalent matrix of the iterative scheme for the adjoint problem is now applied to obtain an adjoint Runge-Kutta method (also explained in [33]), an adjoint residual smoothing and an adjoint multigrid method.

B.5.1. Adjoint Runge-Kutta scheme

If we intend to iterate using a Runge-Kutta scheme, with partial updates of the viscous terms [55], defining $\widetilde{\mathbf{u}}^{(m)} = \mathbf{u}^{(m)} - \mathbf{u}^n$, and $\widetilde{\mathbf{d}}^{(m)} = \mathbf{d}^{(m)} - D\mathbf{u}^n$, the Runge-Kutta scheme can be expressed as:

$$\begin{aligned}
\widetilde{\mathbf{d}}^{(0)} &= -D\mathbf{u}^n \\
\widetilde{\mathbf{u}}^{(0)} &= 0 \\
\left.\begin{aligned}
\widetilde{\mathbf{d}}^{(m)} &= \beta_m D\widetilde{\mathbf{u}}^{(m-1)} + (1-\beta_m)\,\widetilde{\mathbf{d}}^{(m-1)} \\
\widetilde{\mathbf{u}}^{(m)} &= \alpha_m P \left[(\mathbf{f} - L\mathbf{u}^n) - C\widetilde{\mathbf{u}}^{(m-1)} - \widetilde{\mathbf{d}}^{(m)} \right]
\end{aligned}\right\} & \quad m = 1, \ldots, M \\
\mathbf{u}^{n+1} &= \mathbf{u}^n + \widetilde{\mathbf{u}}^{(M)}
\end{aligned}$$

being C and D the matrices that represent the convective and viscous terms, respectively, and P a preconditioning matrix. The Runge-Kutta scheme can also be written with matrices,

$$[RK]\begin{bmatrix} \widetilde{\mathbf{u}}^{(1)} \\ \widetilde{\mathbf{d}}^{(2)} \\ \widetilde{\mathbf{u}}^{(2)} \\ \vdots \\ \widetilde{\mathbf{d}}^{(M)} \\ \widetilde{\mathbf{u}}^{(M)} \end{bmatrix} = \begin{bmatrix} \alpha_1 P \\ 0 \\ \alpha_2 P \\ \vdots \\ 0 \\ \alpha_M P \end{bmatrix} [\mathbf{f} - L\mathbf{u}^n], \tag{B.13}$$

being

$$[RK] = \begin{bmatrix} I & 0 & \dots & & & \dots & 0 \\ -\beta_2 D & I & 0 & \dots & & \dots & 0 \\ \alpha_2 PC & \alpha_2 P & I & 0 & \dots & \dots & 0 \\ 0 & -(1-\beta_3) & -\beta_3 D & I & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & -(1-\beta_M) & -\beta_M D & I & 0 \\ 0 & & & & \alpha_M PC & \alpha_M P & I \end{bmatrix}$$

By solving the system of equations (B.13), the $\widetilde{\mathbf{u}}^{(M)}$ variable can be expressed as a product of the matrix R and the residual vector $\mathbf{f} - L\mathbf{u}^n$,

$$\widetilde{\mathbf{u}}^{(M)} = R\,(\mathbf{f} - L\mathbf{u}^n) = \begin{bmatrix} 0 & 0 & \dots & I \end{bmatrix} [RK]^{-1} \begin{bmatrix} \alpha_1 P \\ 0 \\ \alpha_2 P \\ \vdots \\ 0 \\ \alpha_M P \end{bmatrix} (\mathbf{f} - L\mathbf{u}^n)$$

By identifying the preceding equation with Eq. (B.10), and making use of the result obtained in Eq. (B.12), the matrix of the adjoint iterative scheme is obtained:

$$R^T = \begin{bmatrix} \alpha_1 P^T & 0 & \alpha_2 P^T & 0 & \dots & \alpha_M P^T \end{bmatrix} \left([RK]^T \right)^{-1} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ I \end{bmatrix},$$

which produces an adjoint Runge-Kutta scheme

$$\boldsymbol{\psi}^n = \boldsymbol{\psi}^{n+1} + \sum_{m=1}^{M} \alpha_m P^T \widetilde{\mathbf{w}}^{(m)}$$

where $\widetilde{\mathbf{w}}^{(m)}$ is the solution of the system of equations

$$[RK]^T \begin{bmatrix} \widetilde{\mathbf{w}}^{(0)} \\ \vdots \\ \widetilde{\mathbf{w}}^{(M)} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ I \end{bmatrix}$$

Defining $\widetilde{\boldsymbol{\psi}}^{(m)} = P^T \widetilde{\mathbf{w}}^{(m)}$ we obtain the definition of the adjoint Runge-Kutta scheme with partial updates of the viscous terms

$$\begin{aligned} \widetilde{\boldsymbol{\psi}}^{(M)} &= P^T \left(\mathbf{g} - L^T \boldsymbol{\psi}^{n+1} \right) \\ \widetilde{\mathbf{d}}^{(M)} &= -\alpha_M \widetilde{\boldsymbol{\psi}}^{(M)} \\ \left.\begin{aligned} \widetilde{\boldsymbol{\psi}}^{(m)} &= P^T \left(-\alpha_{m+1} C^T \widetilde{\boldsymbol{\psi}}^{(m+1)} + \beta_{m+1} D^T \widetilde{\mathbf{d}}^{(m+1)} \right) \\ \widetilde{\mathbf{d}}^{(m)} &= -\alpha_m \widetilde{\boldsymbol{\psi}}^{(m)} + (1-\beta_m)\, \widetilde{\mathbf{d}}^{(m+1)} \end{aligned} \right\} & \quad m = M-1, \dots, 1 \\ \boldsymbol{\psi}^n &= \boldsymbol{\psi}^{n+1} + \sum_{m=1}^{M} \alpha_m \ \widetilde{\boldsymbol{\psi}}^{(m)} \end{aligned}$$

B.5.2. Adjoint implicit residual smoothing

The implicit residual smoothing is commonly used to expand the stability region of the Runge-Kutta schemes [41]. It consists in adding a weighted pseudo-Laplacian to the residual $\mathbf{r} = \mathbf{f} - L\mathbf{u}$, yielding a modified residual

$$\bar{\mathbf{r}} = \mathbf{r} + \varepsilon \, [\#ed] \, [\mathbf{pL}] \, \bar{\mathbf{r}}$$

The resolution of this equation is costly, hence the modified residual is approximated by performing a small number of iterations using a weighted Jacobi method [37], yielding the following iterative scheme:

$$\bar{\mathbf{r}}_i^{new} = \frac{1}{1+\varepsilon \#ed_i}\left(\mathbf{r} + \sum_{k=1}^{\#ed_i} \bar{\mathbf{r}}_k^{old}\right)$$

where $\bar{\mathbf{r}}_k$ is the residual of the neighbour nodes of the node i. Therefore, the modified residual can be expressed as

$$\bar{\mathbf{r}} = [R_{RS}]\,\mathbf{r},$$

where

$$[R_{RS}] = \left[\frac{1}{\{1+\varepsilon\,(\#ed)\}}\right]\left[I + \varepsilon \sum_{mn=1}^{Nedges}\begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}\left[\frac{1}{\{1+\varepsilon\,(\#ed)\}}\right]\left\{I + \varepsilon \sum_{kl=1}^{Nedges}\begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}[\cdots]\right\}\right],$$

which transposed provides the expression of the adjoint implicit residual smoothing matrix,

$$[R_{RS}]^T = \left[I + \left\{[\cdots]\,\varepsilon \sum_{kl=1}^{Nedges}\begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix} + I\right\}\left[\frac{1}{\{1+\varepsilon\,(\#ed)\}}\right]\varepsilon \sum_{mn=1}^{Nedges}\begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}\right]\left[\frac{1}{\{1+\varepsilon\,(\#ed)\}}\right]$$

B.5.3. Adjoint multigrid

The matrix of the multigrid iterative scheme$[R_{MG}]$ can be expressed as a product of three steps,

$$[R_{MG}] = \left[I_h^H\right]\,[E]\,\left[I_H^h\right],$$

where $\left[I_H^h\right]$ stands for the restriction of the fine grid residuals into the coarse ones, $[E]$ represents the evolution of the variables in the coarse grid, which includes the Runge-Kutta steps in the coarse grid level and the use of coarser grid levels for further smoothing, and $\left[I_h^H\right]$ the prolongation of the correction back to the fine grid. The transposed matrix will then be

$$[R_{MG}]^T = \left[I_H^h\right]^T\,[E]^T\,\left[I_h^H\right]^T$$

The adjoint restriction operator turns out to be the transposed prolongation operator and vice versa, the adjoint prolongation operator is the transposed restriction operator.
In our agglomeration multigrid strategy, the coarse grid fluxes, $\mathbf{F}_H$, are obtained from the fine grid ones, $\mathbf{F}_h$, with the following formula

$$\mathbf{F}_{k,H} = \frac{1}{\vartheta_{k,H}} \sum_{i=1}^{\vartheta_{i,h} \in \vartheta_{k,H}} \vartheta_{i,h} \cdot \mathbf{F}_{i,h}$$

where ϑ_h and ϑ_H represent the fine and coarse grid node volumes, respectively. The operator is represented by a $N_{nodeCoarse} \times N_{nodeFine}$ matrix

$$\left[I_H^h\right] = \begin{bmatrix} \cdots & \vartheta_{m,h}/\vartheta_{j,H} & \cdots & \vartheta_{n,h}/\vartheta_{j,H} \\ \vartheta_{p,h}/\vartheta_{k,H} & \cdots & \vartheta_{q,h}/\vartheta_{k,H} & \cdots \end{bmatrix}$$

By transposing it, the matrix that represents the adjoint prolongation operator is obtained,

$$\left[I_H^h\right]^T = \begin{bmatrix} \vdots & \vartheta_{p,h}/\vartheta_{k,H} \\ \vartheta_{m,h}/\vartheta_{j,H} & \vdots \\ \vdots & \vartheta_{q,h}/\vartheta_{k,H} \\ \vartheta_{n,h}/\vartheta_{j,H} & \vdots \end{bmatrix},$$

which yields a relation between adjoint coarse and fine grid fluxes

$$\mathbf{F}_{i,h}^T = \frac{\vartheta_{i,h}}{\vartheta_{k,H}} \mathbf{F}_{k,H}^T, \qquad \forall \vartheta_{i,h} \in \vartheta_{k,H}$$

The prolongation operator is

$$\mathbf{F}_{i,h} = \mathbf{F}_{k,H}, \qquad \forall \vartheta_{i,h} \in \vartheta_{k,H},$$

thus giving a $N_{nodeFine} \times N_{nodeCoarse}$ prolongation matrix

$$\left[I_h^H\right] = \begin{bmatrix} \vdots & 1 \\ 1 & \vdots \\ \vdots & 1 \\ 1 & \vdots \end{bmatrix}$$

The transposed matrix of $\left[I_h^H\right]$ yields the matrix of the adjoint restriction operator

$$\left[I_h^H\right]^T = \begin{bmatrix} \cdots & 1 & \cdots & 1 \\ 1 & \cdots & 1 & \cdots \end{bmatrix},$$

that can be expressed as

$$\mathbf{F}_{k,H}^{T} = \sum_{i=1}^{\vartheta_{i,h} \in \vartheta_{k,H}} \mathbf{F}_{i,h}^{T}$$

C. $SKEH$ and penalty function derivatives

Before solving the adjoint Navier-Stokes equations (Eq. (8.6)), we must evaluate the derivatives of the cost function with respect to the conservative variables, $\partial f/\partial \mathbf{U}_j$. This is done analytically. We also obtain the variation of the cost function with respect to variations in the geometric parameters, $\partial f/\partial \boldsymbol{\varphi}_k$, which is necessary to evaluate the cost function gradient (Eq. (8.7)).

In chapter 9, we have performed a gradient-based optimisation to design the non-axisymmetric end walls of a turbine row. The cost function we have minimised is the $SKEH$, and we have also used a swirl angle based penalty function to control the massive separation of the flow downstream the non-axisymmetric perturbations. In this appendix, we obtain the derivatives of these functions.

C.1. $SKEH$ function derivatives

The $SKEH_i$ function of Eq. (9.3), obtained for a mesh point i that belongs to the plane cut where the $SKEH$ is evaluated, is expressed as:

$$SKEH_i = \frac{(\mathbf{v}_i - \mathbf{v}_{pi})^2 \, |\mathbf{v}_i \cdot \boldsymbol{\omega}_i|}{\mathbf{v}_{exit}^4/\ell_c},$$

where

$$\boldsymbol{\omega}_i \;=\; \nabla \times \mathbf{v}_i$$

is the vorticity vector, and

$$\mathbf{v}_{pi} \;=\; \frac{\mathbf{v}_i \cdot \mathbf{v}_m}{\mathbf{v}_m^2}\mathbf{v}_m, \tag{C.1}$$

is the projection of the local velocity over the circumferentially mass averaged velocity $\mathbf{v}_m$ at a

specified radial band, that is expressed as:

$$\mathbf{v}_m = \frac{\sum_{i=1}^{n_\theta} \rho_i \left(\mathbf{v}_i \cdot \mathbf{n}\sigma_i\right) \mathbf{v}_i}{\sum_{i=1}^{n_\theta} \rho_i \left(\mathbf{v}_i \cdot \mathbf{n}\sigma_i\right)}, \tag{C.2}$$

where n_θ is the number of points in the radial band where the mass average is performed. First, we obtain the derivatives of the mass averaged velocity. If we define g_θ as the mass flow over the radial band of interest:

$$g_\theta = \sum_{i=1}^{n_\theta} \rho_i \left(\mathbf{v}_i \cdot \mathbf{n}\sigma_i\right),$$

then the derivatives of Eq. (C.2) with respect to variations in the geometric parameters and velocity components are expressed as:

$$\begin{aligned}
\frac{\partial \mathbf{v}_m}{\partial \boldsymbol{\varphi}_k} &= \frac{1}{g_\theta}\left\{\sum_{i=1}^{n_\theta} \rho_i \left(\mathbf{v}_i \cdot \frac{\partial\left(\mathbf{n}\sigma_i\right)}{\partial \boldsymbol{\varphi}_k}\right)\mathbf{v}_i - \mathbf{v}_m \cdot \sum_{i=1}^{n_\theta} \rho_i \left(\mathbf{v}_i \cdot \frac{\partial\left(\mathbf{n}\sigma_i\right)}{\partial \boldsymbol{\varphi}_k}\right)\right\} \\
\frac{\partial \mathbf{v}_m}{\partial \rho_i} &= \frac{\mathbf{v}_i \cdot \mathbf{n}\sigma_i}{g_\theta}\left(\mathbf{v}_i - \mathbf{v}_m\right) \\
\frac{\partial \mathbf{v}_m}{\partial u_i} &= \frac{\rho_i}{g_\theta}\left\{dS_{xi}\left(\mathbf{v}_i - \mathbf{v}_m\right) + \left(\mathbf{v}_i \cdot \mathbf{n}\sigma_i\right)\begin{bmatrix}1\\0\\0\end{bmatrix}\right\} \\
\frac{\partial \mathbf{v}_m}{\partial v_i} &= \frac{\rho_i}{g_\theta}\left\{dS_{yi}\left(\mathbf{v}_i - \mathbf{v}_m\right) + \left(\mathbf{v}_i \cdot \mathbf{n}\sigma_i\right)\begin{bmatrix}0\\\cos\theta_i\\\sin\theta_i\end{bmatrix}\right\} \\
\frac{\partial \mathbf{v}_m}{\partial w_i} &= \frac{\rho_i}{g_\theta}\left\{dS_{zi}\left(\mathbf{v}_i - \mathbf{v}_m\right) + \left(\mathbf{v}_i \cdot \mathbf{n}\sigma_i\right)\begin{bmatrix}0\\-\sin\theta_i\\\cos\theta_i\end{bmatrix}\right\}
\end{aligned}$$

where $\theta_i = \arctan\left(y/z\right)$. Making use of these expressions, the derivatives of the projected velocity of Eq. (C.1) are:

$$\begin{aligned}
\frac{\partial \mathbf{v}_{pi}}{\partial \boldsymbol{\varphi}_k} &= \frac{1}{\mathbf{v}_m^2}\left[\left(\mathbf{v}_i \cdot \mathbf{v}_m\right)\frac{\partial \mathbf{v}_m}{\partial \boldsymbol{\varphi}_k} + \left(\mathbf{v}_i \cdot \frac{\partial \mathbf{v}_m}{\partial \boldsymbol{\varphi}_k}\right)\mathbf{v}_m - 2\left(\mathbf{v}_m \cdot \frac{\partial \mathbf{v}_m}{\partial \boldsymbol{\varphi}_k}\right)\mathbf{v}_{pi}\right] \\
\frac{\partial \mathbf{v}_{pi}}{\partial \rho_i} &= \frac{\mathbf{v}_i \cdot \mathbf{v}_m}{\mathbf{v}_m^2}\left[\frac{\partial \mathbf{v}_m}{\partial \rho_i} - \frac{2}{\mathbf{v}_m^2}\left(\mathbf{v}_m \cdot \frac{\partial \mathbf{v}_m}{\partial \rho_i}\right)\mathbf{v}_m\right] + \frac{1}{\mathbf{v}_m^2}\left(\mathbf{v}_i \cdot \frac{\partial \mathbf{v}_m}{\partial \rho_i}\right)\mathbf{v}_m \\
\frac{\partial \mathbf{v}_{pi}}{\partial u_i} &= \frac{\mathbf{v}_i \cdot \mathbf{v}_m}{\mathbf{v}_m^2}\left[\frac{\partial \mathbf{v}_m}{\partial u_i} - \frac{2}{\mathbf{v}_m^2}\left(\mathbf{v}_m \cdot \frac{\partial \mathbf{v}_m}{\partial u_i}\right)\mathbf{v}_m\right] + \frac{1}{\mathbf{v}_m^2}\left[\left(\mathbf{v}_i \cdot \frac{\partial \mathbf{v}_m}{\partial u_i}\right) + u_m\right]\mathbf{v}_m
\end{aligned}$$

$$\frac{\partial \mathbf{v}_{pi}}{\partial v_i} = \frac{\mathbf{v}_i \cdot \mathbf{v}_m}{\mathbf{v}_m^2}\left[\frac{\partial \mathbf{v}_m}{\partial v_i} - \frac{2}{\mathbf{v}_m^2}\left(\mathbf{v}_m \cdot \frac{\partial \mathbf{v}_m}{\partial v_i}\right)\mathbf{v}_m\right] + \frac{1}{\mathbf{v}_m^2}\left[\left(\mathbf{v}_i \cdot \frac{\partial \mathbf{v}_m}{\partial v_i}\right) + v_m\right]\mathbf{v}_m$$
$$\frac{\partial \mathbf{v}_{pi}}{\partial w_i} = \frac{\mathbf{v}_i \cdot \mathbf{v}_m}{\mathbf{v}_m^2}\left[\frac{\partial \mathbf{v}_m}{\partial w_i} - \frac{2}{\mathbf{v}_m^2}\left(\mathbf{v}_m \cdot \frac{\partial \mathbf{v}_m}{\partial w_i}\right)\mathbf{v}_m\right] + \frac{1}{\mathbf{v}_m^2}\left[\left(\mathbf{v}_i \cdot \frac{\partial \mathbf{v}_m}{\partial w_i}\right) + w_m\right]\mathbf{v}_m$$

And then the $SKEH_i$ derivatives yield:

$$\frac{\partial SKEH_i}{\partial \boldsymbol{\varphi}_k} = -\frac{2}{\mathbf{v}_{exit}^4/\ell_c}\left(\mathbf{v}_i - \mathbf{v}_{pi}\right) \cdot \frac{\partial \mathbf{v}_{pi}}{\partial \boldsymbol{\varphi}_k}\left|\mathbf{v}_i \cdot \boldsymbol{\omega}_i\right|$$
$$\frac{\partial SKEH_i}{\partial \rho_i} = -\frac{2}{\mathbf{v}_{exit}^4/\ell_c}\left(\mathbf{v}_i - \mathbf{v}_{pi}\right) \cdot \frac{\partial \mathbf{v}_{pi}}{\partial \rho_i}\left|\mathbf{v}_i \cdot \boldsymbol{\omega}_i\right|$$
$$\frac{\partial SKEH_i}{\partial u_i} = \frac{1}{\mathbf{v}_{exit}^4/\ell_c}\left[2\left(\mathbf{v}_i - \mathbf{v}_{pi}\right) \cdot \left\{\begin{bmatrix}1\\0\\0\end{bmatrix} - \frac{\partial \mathbf{v}_{pi}}{\partial u_i}\right\}\left|\mathbf{v}_i \cdot \boldsymbol{\omega}_i\right| + \left|\mathbf{v}_i - \mathbf{v}_{pi}\right|^2 sig\left(\mathbf{v}_i \cdot \boldsymbol{\omega}_i\right)\omega_{xi}\right]$$
$$\frac{\partial SKEH_i}{\partial v_i} = \frac{1}{\mathbf{v}_{exit}^4/\ell_c}\left[2\left(\mathbf{v}_i - \mathbf{v}_{pi}\right) \cdot \left\{\begin{bmatrix}0\\\cos\theta_i\\\sin\theta_i\end{bmatrix} - \frac{\partial \mathbf{v}_{pi}}{\partial v_i}\right\}\left|\mathbf{v}_i \cdot \boldsymbol{\omega}_i\right| + \left|\mathbf{v}_i - \mathbf{v}_{pi}\right|^2 sig\left(\mathbf{v}_i \cdot \boldsymbol{\omega}_i\right)\omega_{yi}\right]$$
$$\frac{\partial SKEH_i}{\partial w_i} = \frac{1}{\mathbf{v}_{exit}^4/\ell_c}\left[2\left(\mathbf{v}_i - \mathbf{v}_{pi}\right) \cdot \left\{\begin{bmatrix}0\\-\sin\theta_i\\\cos\theta_i\end{bmatrix} - \frac{\partial \mathbf{v}_{pi}}{\partial w_i}\right\}\left|\mathbf{v}_i \cdot \boldsymbol{\omega}_i\right| + \left|\mathbf{v}_i - \mathbf{v}_{pi}\right|^2 sig\left(\mathbf{v}_i \cdot \boldsymbol{\omega}_i\right)\omega_{zi}\right]$$

Besides, the $SKEH_i$ expression also depends on the exit patch mass averaged velocity v_{exit}, that is computed making use of the areas associated to the boundary edges:

$$v_{exit} = \frac{\sum_{i=1}^{n_{out}}\left(\rho_{ki}\left|\mathbf{v}_{ki}\right|\mathbf{v}_{ki} + \rho_{kj}\left|\mathbf{v}_{kj}\right|\mathbf{v}_{kj}\right) \cdot \mathbf{n}\sigma_i}{\sum_{i=1}^{n_{out}}\left(\rho_{ki}\mathbf{v}_{ki} + \rho_{kj}\mathbf{v}_{nj}\right) \cdot \mathbf{n}\sigma_i},$$

where ki and kj are the nodes that conform the boundary edge i and n_{out} stands for the number of edges that conform the exit patch. We define the mass flow over the exit patch as a sum over the boundary edges that belong to it:

$$g_{out} = \sum_{i=1}^{n_{out}}\left(\rho_{ki}\mathbf{v}_{ki} + \rho_{kj}\mathbf{v}_{kj}\right) \cdot \mathbf{n}\sigma_i,$$

The derivatives of v_{exit} depend solely on the exit patch nodes, and we assume that the changes in the geometry do not affect the exit patch, thus the variation with respect to this parameter

is omitted:

$$\begin{aligned}
\frac{\partial v_{exit}}{\partial \rho_{ki}} &= \frac{1}{g_{out}} \sum_{edge \in E_{ki}} \left(|\mathbf{v}_{ki}| - v_{exit}\right) \left(\mathbf{v}_{ki} \cdot \mathbf{n}\sigma_i\right) \\
\frac{\partial v_{exit}}{\partial u_{ki}} &= \frac{1}{g_{out}} \sum_{edge \in E_{ki}} \left[\rho_{ki} \left(|\mathbf{v}_{ki}| - v_{exit}\right) dS_{xi} + \rho_{ki} \frac{u_{ki}}{|\mathbf{v}_{ki}|} \left(\mathbf{v}_{ki} \cdot \mathbf{n}\sigma_i\right)\right] \\
\frac{\partial v_{exit}}{\partial v_{ki}} &= \frac{1}{g_{out}} \sum_{edge \in E_{ki}} \left[\rho_{ki} \left(|\mathbf{v}_{ki}| - v_{exit}\right) dS_{yi} + \rho_{ki} \frac{v_{ki}}{|\mathbf{v}_{ki}|} \left(\mathbf{v}_{ki} \cdot \mathbf{n}\sigma_i\right)\right] \\
\frac{\partial v_{exit}}{\partial w_{ki}} &= \frac{1}{g_{out}} \sum_{edge \in E_{ki}} \left[\rho_{ki} \left(|\mathbf{v}_{ki}| - v_{exit}\right) dS_{zi} + \rho_{ki} \frac{w_{ki}}{|\mathbf{v}_{ki}|} \left(\mathbf{v}_{ki} \cdot \mathbf{n}\sigma_i\right)\right]
\end{aligned}$$

where E_{ki} are the group of edges that contain the ki node. With these equations, the variation of the $SKEH_i$ with respect to the exit variables is:

$$\frac{\partial SKEH_i}{\partial \xi_{iout}} = -\frac{4}{v_{exit}} SKEH_i \frac{\partial v_{exit}}{\partial \xi_{iout}}$$

In this expression, ξ_{iout} is either ρ_{ni}, u_{ni}, v_{ni} or w_{ni}.

The cost function used for the optimisation problem is the mass averaged $SKEH$ over the plane cut:

$$\overline{SKEH} = \frac{\sum_{i=1}^{n_{cp}} \rho_i \left(\mathbf{v}_i \cdot \mathbf{n}\sigma_i\right) SKEH_i}{\sum_{i=1}^{n_{cp}} \rho_i \left(\mathbf{v}_i \cdot \mathbf{n}\sigma_i\right)},$$

where n_{cp} is the number of mesh points contained in that plane cut. The variation of the cost function with respect to changes in the geometric parameters is:

$$\begin{aligned}
\frac{d\overline{SKEH}}{d\boldsymbol{\varphi}_k} &= \frac{\partial \overline{SKEH}}{\partial \boldsymbol{\varphi}_k} + \sum_{i=1}^{n_{cp}} \left\{ \frac{\partial \overline{SKEH}}{\partial \rho_i} \frac{\partial \rho_i}{\partial \boldsymbol{\varphi}_k} + \frac{\partial \overline{SKEH}}{\partial u_i} \frac{\partial u_i}{\partial \boldsymbol{\varphi}_k} + \frac{\partial \overline{SKEH}}{\partial v_i} \frac{\partial v_i}{\partial \boldsymbol{\varphi}_k} + \frac{\partial \overline{SKEH}}{\partial w_i} \frac{\partial w_i}{\partial \boldsymbol{\varphi}_k} \right\} \\
&+ \sum_{i=1}^{n_{out}} \left\{ \frac{\partial \overline{SKEH}}{\partial \rho_{iout}} \frac{\partial \rho_{iout}}{\partial \boldsymbol{\varphi}_k} + \frac{\partial \overline{SKEH}}{\partial u_{iout}} \frac{\partial u_{iout}}{\partial \boldsymbol{\varphi}_k} + \frac{\partial \overline{SKEH}}{\partial v_{iout}} \frac{\partial v_{iout}}{\partial \boldsymbol{\varphi}_k} + \frac{\partial \overline{SKEH}}{\partial w_{iout}} \frac{\partial w_{iout}}{\partial \boldsymbol{\varphi}_k} \right\},
\end{aligned}$$

where

$$\begin{aligned}
\frac{\partial \overline{SKEH}}{\partial \boldsymbol{\varphi}_k} &= \frac{1}{g_{cp}} \left[\sum_{i=1}^{n_{cp}} \rho_i \left(\mathbf{v}_i \cdot \frac{\partial \left(\mathbf{n}\sigma_i\right)}{\partial \boldsymbol{\varphi}_k} \right) SKEH_i - \overline{SKEH} \sum_{i=1}^{n_{cp}} \rho_i \left(\mathbf{v}_i \cdot \frac{\partial \left(\mathbf{n}\sigma_i\right)}{\partial \boldsymbol{\varphi}_k} \right) + \right. \\
&+ \left. \sum_{i=1}^{n_{cp}} \rho_i \left(\mathbf{v}_i \cdot \frac{\partial \left(\mathbf{n}\sigma_i\right)}{\partial \boldsymbol{\varphi}_k} \right) \frac{\partial SKEH_i}{\partial \boldsymbol{\varphi}_k} \right] \\
\frac{\partial \overline{SKEH}}{\partial \rho_i} &= \frac{1}{g_{cp}} \left[\left(\mathbf{v}_i \cdot \mathbf{n}\sigma_i\right) \left(SKEH_i - \overline{SKEH}\right) + \rho_i \left(\mathbf{v}_i \cdot \mathbf{n}\sigma_i\right) \frac{\partial SKEH_i}{\partial \rho_i} \right]
\end{aligned}$$

$$\begin{aligned}
\frac{\partial\overline{SKEH}}{\partial u_i} &= \frac{\rho_i}{g_{cp}}\left[dS_{xi}\left(SKEH_i-\overline{SKEH}\right)+\left(\mathbf{v}_i\cdot\mathbf{n}\sigma_i\right)\frac{\partial SKEH_i}{\partial u_i}\right]\\
\frac{\partial\overline{SKEH}}{\partial v_i} &= \frac{\rho_i}{g_{cp}}\left[dS_{yi}\left(SKEH_i-\overline{SKEH}\right)+\left(\mathbf{v}_i\cdot\mathbf{n}\sigma_i\right)\frac{\partial SKEH_i}{\partial v_i}\right]\\
\frac{\partial\overline{SKEH}}{\partial w_i} &= \frac{\rho_i}{g_{cp}}\left[dS_{zi}\left(SKEH_i-\overline{SKEH}\right)+\left(\mathbf{v}_i\cdot\mathbf{n}\sigma_i\right)\frac{\partial SKEH_i}{\partial w_i}\right]\\
\frac{\partial\overline{SKEH}}{\partial\xi_{iout}} &= -\frac{4}{v_{exit}}\overline{SKEH}\frac{\partial v_{exit}}{\partial\xi_{iout}}
\end{aligned}$$

In these expressions,

$$g_{cp}=\sum_{i=1}^{n_{cp}}\rho_i\left(\mathbf{v}_i\cdot\mathbf{n}\sigma_i\right)$$

is the mass flow over the plane cut where the $SKEH$ is evaluated.

C.2. Penalty function derivatives

As mentioned in sec. 9.2, the use of the $SKEH$ alone as a cost function for the optimisation problem is not sufficient to detect phenomena, like massive separation of the flow, that can be controlled by adding a penalty function to the $SKEH$. That function takes into account the deviations of the optimised case swirl angle from the initial geometry in the same axial plane cut where the $SKEH$ is evaluated. The proposed function is:

$$F_p=f_1\cdot e^{-\frac{\overline{\beta}_s-\beta_t+f_2}{f_3}},$$

where $\overline{\beta}_s$ is the mass averaged slope swirl angle, β_t is the mass averaged initial slope swirl angle, and f_1, f_2 y f_3 are constant factors that adjust the shape of the function, thus making it rise when the overturning grows. Note that the function only penalises the overturning, since it is considered that diminishing the flow turning in the secondary flow area improves the downstream flow distribution. The derivative of that function is:

$$dF_p=-\frac{F_p}{f_3}d\overline{\beta}_s$$

The mass averaged slope swirl angle in the plane cut is

$$\overline{\beta}_s=\frac{\sum_{i=1}^{n_{cp}}\rho_i\left(\mathbf{v}_i\cdot\mathbf{n}\sigma_i\right)\beta_{si}}{\sum_{i=1}^{n_{cp}}\rho_i\left(\mathbf{v}_i\cdot\mathbf{n}\sigma_i\right)}\cdot f\left(r\right),$$

where

$$\beta_{si} = \arcsin \frac{v_{\theta i}}{|\mathbf{v}_i|},$$

and $f(r)$ is a radial weighting function. This function is introduced because the angle values we are interested in are these of the secondary flow zone and next to the walls, hence an average over all span misrepresents the angle deviation in the zones of interest. Therefore, $f(r) = 1$ in these zones, and $f(r) = 0$ far from the end-walls. The slope swirl angle derivatives are:

$$\begin{aligned}
\frac{\partial \overline{\beta}_s}{\partial \boldsymbol{\varphi}_k} &= \frac{1}{g_{cp}} \left\{ \sum_{i=1}^{n_{cp}} \rho_i \left(\mathbf{v}_i \cdot \frac{\partial (\mathbf{n}\sigma_i)}{\partial \boldsymbol{\varphi}_k} \right) \beta_{si} - \overline{\beta}_s \cdot \sum_{i=1}^{n_{cp}} \rho_i \left(\mathbf{v}_i \cdot \frac{\partial (\mathbf{n}\sigma_i)}{\partial \boldsymbol{\varphi}_k} \right) \right\} \cdot f(r_i) \\
\frac{\partial \overline{\beta}_s}{\partial \rho_i} &= \frac{\mathbf{v}_i \cdot \mathbf{n}\sigma_i}{g_{cp}} \left(\beta_{si} - \overline{\beta}_s \right) \cdot f(r_i) \\
\frac{\partial \overline{\beta}_s}{\partial u_i} &= \frac{\rho_i}{g_{cp}} \left\{ dS_{xi} \left(\beta_{si} - \overline{\beta}_s \right) + (\mathbf{v}_i \cdot \mathbf{n}\sigma_i) \frac{\partial \beta_{si}}{\partial u_i} \right\} \cdot f(r_i) \\
\frac{\partial \overline{\beta}_s}{\partial v_i} &= \frac{\rho_i}{g_{cp}} \left\{ dS_{yi} \left(\beta_{si} - \overline{\beta}_s \right) + (\mathbf{v}_i \cdot \mathbf{n}\sigma_i) \frac{\partial \beta_{si}}{\partial v_i} \right\} \cdot f(r_i) \\
\frac{\partial \overline{\beta}_s}{\partial w_i} &= \frac{\rho_i}{g_{cp}} \left\{ dS_{zi} \left(\beta_{si} - \overline{\beta}_s \right) + (\mathbf{v}_i \cdot \mathbf{n}\sigma_i) \frac{\partial \beta_{si}}{\partial w_i} \right\} \cdot f(r_i)
\end{aligned}$$

where

$$\begin{aligned}
\frac{\partial \beta_{si}}{\partial u_i} &= -\frac{\lambda}{|\mathbf{v}_i| \sqrt{1-\lambda^2}} \frac{u_i}{|\mathbf{v}_i|} \\
\frac{\partial \beta_{si}}{\partial v_i} &= \frac{1}{|\mathbf{v}_i| \sqrt{1-\lambda^2}} \left(\cos\theta_i - \lambda \frac{v_i}{|\mathbf{v}_i|} \right) \\
\frac{\partial \beta_{si}}{\partial w_i} &= -\frac{1}{|\mathbf{v}_i| \sqrt{1-\lambda^2}} \left(\sin\theta_i + \lambda \frac{w_i}{|\mathbf{v}_i|} \right)
\end{aligned}$$

being $\lambda = v_{\theta i} / |\mathbf{v}_i|$